Ontology Driven generation of tools for building and navigating knowledge bases

by

Christian Henriksen
&
Tommy Johannessen

Masters Thesis in
Information and Communication Technology

Agder University College
Faculty of Engineering and Science

Grimstad, June 2004

Abstract

Throughout the work of this thesis we have considered how to apply usability engineering to multiple different projects using their own ontologies. Reviews on existing applications similar to the Topic Map were made. The reviews gave us valuable information on what kind of usability aspects and presentation techniques other model driven applications use.

Software projects are often built manually by planning and programming the application from scratch. Needing more than one project with similarities, auto-generation can be a viable solution. However, auto-generation does in general not provide sufficient usability to end products. In this thesis we have investigated how to develop an application with auto-generation, including the process of applying usability and utility engineering.

All projects, whether it is manually or auto generated, needs final customization to fit the customer's, and the end user's demands. Therefore we had to find a way to improve the output from the auto-generation, and if possible provide an understandable layer to make the final implementation of usability and utility higher. A prototype showing how the auto-generation application works was made.

The ontologies defined in this thesis use Topic Maps. These ontologies are then used to auto generate the application. Developers should be able to customise the application, and therefore customisable templates are generated along with the application. Using the Topic Map concepts in these templates can confuse the developers. To make the templates more logical for the developers, we decided to make a layer - a reflective API. This layer avoids technology specific terms provided by the Topic Map engine. Methods generated are based on the ontologies, and are therefore more logical for the developers. These methods make it easier to add functionality. The templates are used to apply usability and utility to the applications.

During this thesis we have shown that generating a three layer model can benefit the developer directly, and indirectly the end user since the developer then can spend more time with usability engineering.

Preface

This thesis was written in cooperation with the company InterMedium in Grimstad, as a part of the Master degree in Information and Communication Technology at Agder University College. The work was carried out in the period between January 2004 and May 2004.

We would like to thank our supervisor, Lars Line at Agder University College, Asle Pedersen and Bernt Andre Solheim at InterMedium, for valuable help during the entire process of this thesis.

Grimstad, May 2004

Christian Henriksen

Tommy Johannsesen

Table of Contents

1 Introduction 1
1.1 Background 1
1.2 Thesis definition. 2
1.3 Work description 2
1.4 Report outline 3
2 Theory - Topic Map, Visualization and Usability 4
2.1 Topic Map and structuring of information 4
2.1.1 Topic Map concepts 5
2.1.2 Topic Map Query Language 7
2.1.3 Topic Map Constraint Language 8
2.1.4 Modelling the Topic Map 9
2.2 Visualisation techniques 11
2.2.1 Text-based 11
2.2.2 Tree structure 12
2.2.3 Graph 12
2.2.4 Map 13
2.2.5 3D Visualising 13
2.3 Usability 14
2.3.1 Learn ability 15
2.3.2 Efficiency 15
2.3.3 Memorability 16
2.3.4 Errors 16
2.3.5 Satisfaction 16
3 The InterMedium case 17
3.1 Background information 17
3.2 Describing the InterMedium ontologies 18
3.2.1 The Word of Mouth ontology 18
3.2.2 Other Ontologies 18
4 Review of existing visualization applications 19
4.1 Review Criteria 20
4.1.1 Usability 20
4.1.2 Navigation 20
4.1.3 Edit 21
4.1.4 Import/Export 22
4.1.5 Design and Layout 22
4.1.6 Application specific 22
4.1.7 Criteria table 23
4.2 Selecting applications to review 23
4.3 Review summary 24
4.3.1 Usability 24
4.3.2 Navigation 25
4.3.3 Edit 26
4.3.4 Design and Layout 26
4.3.5 Summary 27
5 The Ideal Solution 28
5.1 The usage of the application 29
5.1.1 The User group 29
5.1.2 The purpose of the application 29
5.2 The Ideal solution – criteria 30
5.2.1 Usability 30
5.2.2 Navigation 31
5.2.3 Edit 32
5.2.4 Import/Export 32
5.2.5 Design and Layout 32
6 The Prototype 34
6.1 The development 35
6.1.1 Three layer model 35
6.1.2 The ontology 35

6.1.3 Presentation layer 36
6.1.4 The web interface 38
6.1.5 Using the Model Interface in other contexts 39
6.2 Customizing the prototype 39
6.2.1 Customizing the content 39
6.2.2 Customizing the Design and Layout 39
7 The prototype vs. the ideal solution 40
7.1 Comparing criteria results 40
7.1.1 Usability 40
7.1.2 Navigation 41
7.1.3 Edit 41
7.1.4 Design and layout 41
7.2 Generation of the user-interface 41
8 Result 42
8.1 The Reviews 42
8.2 Auto generating the reflective API 42
8.3 Generating the Web Application 43
8.4 Assigning usability to the final application 45
9 Discussion 46
9.1 Auto-generation of a user-interface with Topic Maps 46
9.2 Navigating and building the knowledge base 47
9.3 Customise the application and assign usability 48
9.4 Using the reflective API in other contexts 48
10 Further Work 49
11 Conclusion 50
12 References 51

Appendix A	Reviews	
Appendix B	Abstraction of XTM 1.0	
Appendix C	Java documentation	CD-ROM
Appendix D	Java source code	CD-ROM

1 Introduction

1.1 Background

This thesis is performed in cooperation with a local firm called InterMedium. We will start to make a scenario that describes why InterMedium has given this assignment.

A large fictive company called Hoags, selling sports products, wants to enhance the product quality and get feedback from their customers in a new way. Until now, they have used market analysis, user surveys, information from sales, and their accountancy to monitor what their customers feel about Hoags and their products. There are many discussions on the Internet regarding everything from quality on their equipment, sponsoring of individual sport athletes to their latest advertisement campaign. Manual searching through all these pages, sorting out the relevant and usable information is time consuming, and almost impossible. They have now heard of InterMedium, a full service provider of "Competitive Intelligence", providing monitoring of people's actions on the Internet. InterMedium has automated the time consuming component of harvesting the Internet for information. The most essential information is presented in a portal.

InterMedium has advanced agents collecting information from the Internet. Companies like Hoags provide input on what they consider valuable information. Based on these inputs, InterMedium stores the information. They want to provide the relevant data to customers like Hoags based on their inputs. Different customers often have different needs, and relevant information for Hoags might not be so relevant for other companies.

Hoags could search the Internet manually. This is however very time consuming, and it is difficult to find all information without using intelligent search methods.

InterMedium can replace the manual search. They can then provide the result to the customers in such a way that they can easily draw their own conclusions and benefit from the analysis. Sometimes the customers need to add information for various reasons for instance to highlight important aspects of a case.

Semantic layers[1] are used by InterMedium to enhance the meaning of the information collected by the agents and also input from the customers. InterMedium defines an ontology[2] based on ideas about what they consider as important for the customers. The ontology defines the semantic layer which describes the concepts important for InterMedium's customers, and how these concepts are related to each other.

Hoags can have bikes and skates in their assortment. Skates can be discussed in the Hoags Addicts Internet forum[3], administrated and used by Hoags Official Community[4]. Transferring these sentences to ontology would imply; a company can have several products. These products can be discussed in many forums and one forum can just be connected to one community. The ontology provides a set of rules defining important aspects used to keep consistency. The ontology can change for the user and therefore InterMedium needs a system so that they can easily provide a user-interface[5] based on the different ontologies. Ideally the developers at InterMedium could define the ontology and automatically get a user-interface for their customers fulfilling all their needs. Following chapters will look into how this thesis approaches this task.

---

[1] Semantic layers – A layer used to add meaning to information
[2] Ontology is a specification of a conceptualisation. [0]
[3] An online discussion group, newsgroup or other public assembly for discussion
[4] A group of people in some grade sharing the same interests
[5] The specific area of which an user can interact with information for instance stored in a model

1.2 Thesis definition.

Large data repositories often contain huge amounts of information that can be useful for a variety of purposes. Some of these repositories are often heterogeneous and loosely structured information systems. Search for relevant and qualified information can be difficult and time consuming. One approach to more efficient information seeking is standardisation of format, structure and semantics (ontology). Topic Map is a defined standard (ISO13250) and a technology targeting the challenge of semantics and structure of information.

Based on a use-case selected in cooperation with InterMedium, this thesis will review and compare existing applications for building and navigating Topic Maps or similar knowledge structures. A prototype shall be developed, based on the results of the review and the given ontology for the use-case. The objective of the prototype is to explore and demonstrate the feasibility of auto-generation of tools for navigation and building of a knowledge base, represented using Topic Maps.

1.3 Work description

InterMedium uses different ontologies in their projects. Since InterMedium already has good experience with Topic Maps[1] in existing projects, we started to investigate this technology to see how we could describe the ontologies using the Topic Map technology.

It is important for the end users[6] to browse, edit and find interesting information in the model[7]. We investigated and researched visualising principles and usability concepts to see how we could present a model for the users. This research was the foundation for a criteria scheme used to make some reviews on existing model driven applications[8] using similar technology to Topic Maps.

We used the results in the reviews to describe what we considered an ideal solution. It was a requirement from InterMedium that their developers could model their own ontologies and make a user-interface for their customers. Our solution should therefore auto-generate a part of the user-interface.

The ideal solution, the inputs from our supervisor and the time scale of the project were then used to decide what kind of prototype we were to develop. The concepts presented in the ideal solution will be proven in this prototype.

Developers at InterMedium and their customer's needs were taken into consideration when we built the prototype. It was emphasised that customers could easily use the product and extract the knowledge InterMedium presented to them. The developers should also be able to customize the auto generated user-interface suited for the end user.

[6] End users are the users that uses the final application.
[7] In this thesis a model will be referred to as the ontology, and the instances created from it.
[8] Modell Driven Applications (MDA) [13]

1.4 Report outline

This report will start explaining the theory we have used in this thesis. This chapter see how we can use Topic Map to structure the information, and what visualising principles and techniques we can use to present a model. To make the end application suitable for InterMedium's customer, chapter 2 also will look into which usability aspects we should consider. In chapter 3 the case from InterMedium is presented. Chapter 4 will compare and discuss the reviews made on existing model driven applications. The reviews of these are available in Appendix A. In chapter 5, "The Ideal Solution" we have explained what we consider that InterMedium will benefit the most from based on the case. Chapter 6 explains how we have implemented this solution and how the prototype works. Chapter 7 compares the ideal solution with the prototype. The next chapter is the Result chapter. This chapter presents the results from the reviews and the prototype. The last three Chapters, 9, 10 and 11 present Discussion, Further Work and Conclusion.

2 Theory - Topic Map, Visualization and Usability

Whether one should develop an application, construct a building or perform a lecture, knowledge about all these topics are needed. The world consists of indescribable amounts of information, and tomorrow it consists of even more. The knowledge a person has is much information in itself. Most companies have large amounts of data and it is important for them to store this information, so it can easily be retrieved if needed. Before computers were introduced, information was stored in paper documents. Today almost every company uses computers to store their data. In the beginning all the information was stored in computer files. Today many use databases[9] to store information in. It is often very difficult to find the relevant information in large storages. In relation databases you can define relations between data. By doing this it can be easier to find the information you are looking for. But when the relation database gets too large, the data can be difficult to retrieve. Dataware houses are therefore the next step from relation databases. Here you can set the relations between the data in a more structured way.

These solutions do not give a meaning to what role the data plays to each other. By using semantic layers it might be possible to find out how to present the data based on its meaning. In the following chapters we will look into how we can make structured information by giving it Metadata that tells more about the roles and associations. Topic Maps is a way to make use of it. We will look into how we can use the concepts in the Topic Map model to add semantics. We will then see how we this information can be retrieved in a rational way by using visualizing principles and techniques. Usability principles and techniques will be investigated to see how we can make the application understandable for the users

2.1 Topic Map and structuring of information

A map is used to find a destination. The route can be short, fast or long. Walking in the nature you absorb incredible amounts of information. Every stone has a weight, and every tree has a height. All this information is not important to know. The fences, rivers, pitfalls, and gradients are more important. A map often represents the most important surroundings you want to explore. The map itself is of course not equal to the surroundings. It is only a simplification with the most important facts about the environments you need to know to go from A to B. For an ornithologist the map could contain pictures of birds and information about them. There are maps in all kinds, and they are used everyday.

Topic Map is a relatively new standard trying to address semantics in the web. Knowledge presentation, library science, and techniques for linking and addressing have to be used to solve this. It is often said that Topic Maps is building a bridge between information handling, and knowledge presentation.

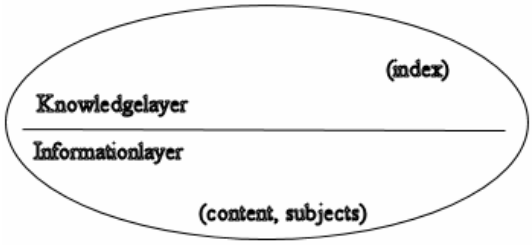


Figure 1 Two layers are here introduced, the knowledge layer, and the Information layer. The knowledge layer can be structured with a Topic Map.

A Topic Map consists of topics providing addresses to information resources. There can be countless relationships between topics in the map. The Topic Map model has a two layer structure and it is divided into the information layer and a knowledge representation level. A topic in the Topic Map can be almost anything you want it to be according to the XTM specification[1]. A resource added to a Topic Map is called a topic and it is a reification of the "real" subject. Topic Map containing only topics would loose some of its strengths. Connections are made between topics when it is relevant.

[9] Database – Organized collection of information in computer systems

Two reificated subjects, a carpenter and a hammer should have a logical connection between each other because the hammer is a tool a carpenter can use. There is also many other associations between these two specific topics, and you have to decide if these connections should be made. It may not be important for a carpenter to know that both he and the hammer are partly made of carbons and share this relation, even though it is true. A powerful property of the associations in a Topic Map is the possibility to give role types to each association. A role type is also a topic. The hammer is used by the carpenter to punch nails. A role type in that association could then be called "punches nails with" between the hammer and the carpenter.

A more detailed and thorough explanation of the different constructs regarding Topic Maps will follow. The XML Topic Maps (XTM) 1.0 standard is a well written standard. To explain this we will first account for the Topic Map concepts, and then see how it is possible to use constraints in the Topic Map standard. We will also look into the possibilities of using query language in Topic Maps. The chapter will sum up with how to use Topic Map concepts to construct a Topic Map for different needs.

2.1.1 Topic Map concepts

Topic

"A topic is a resource that acts as a proxy for some subject; it is the Topic Map system's representation of that subject." - XTM 1.0

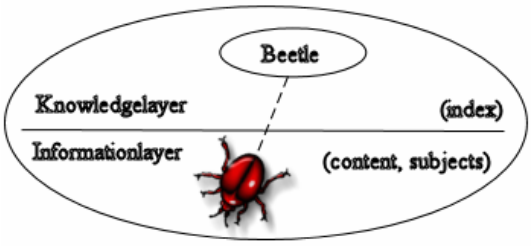


Figure 2 A beetle is a living creature, or maybe an information page about beetles on the Internet. The topic Beetle reifies the Beetle, or the Beetle information page in the knowledge layer.

You are doing a reification of a specific subject when you create a topic. The reification itself is the relationship between the topic and the subject. Characteristics giving a more detailed description to the topic can be assigned as needed.

A Topic Map is consistent when every subject is only represented by one topic in your Topic Map. Non-addressable subjects are subjects existing outside the addressable field in a computer. They have to be addressed indirectly. Examples of such subjects are the person Per Sivle, one of his poems, or the beetle in the picture above. There are countless examples to mention. There also exist resources we can address directly inside a computer called addressable subjects. Example of this could be HIA's homepage, with http://www.hia.no/ as address.

"Topics are what make subjects "real" for the computer system and it's the closest a machine can come to a representation of what is "real" for humans." – XTM 1.0

A subject's identity can be established in two ways; by addressing it directly, or by indicating the subject via a subject indicator. Two topics with the same subject identity should be merged. Cases where the topics should not be merged exist, but they are not common and should be handled separately. A subject indicator is a resource (addressable information resource) indicating the address of a subject.

Topic type

Topic types indicate what the topic is about. It is really a special type of an association. These are on two different Meta levels. You can say the Fruit is part of the ontology and Apple is a part of the model.

This is not necessarily how types are used. We can say that the topic type for fruit is food. Other instances of food could be meat. By building the model like this it is easier to see how meat and food are connected to each other. Next chapter will show that you can also say that apple is food for a person using associations.

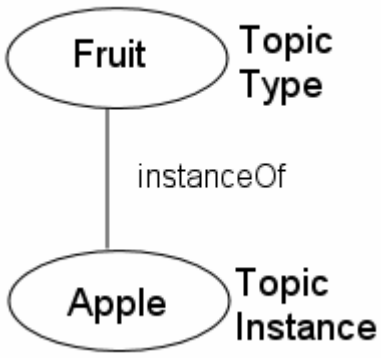


Figure 3 A fruit and an instance of it, an apple. Both are topics in the Topic Map.

Associations

In a relation database relations are built using primary and foreign keys. Relations in a Topic Map are defined with associations between topics.

"An association is a relationship between one or more topics, each of which plays a role as a member of that association. The roles a topic plays in associations are among the characteristics that can be assigned to it and are therefore governed by scope. Each individual association is an instance of a single class of association (also known as an association type) that may or may not be indicated explicitly. The default association type is defined by the "association" published subject." – XTM standard.

By structuring the information with associations in Topic Maps it is possible to give more meaning to the relations between data. A powerful way to describe associations is to assign role types to them. These role types states how the topics interact with each other.

Figure 4 shows that James owns a apple. The figure also says apple is food for James. The topics food and owner are then playing a role in this association. This is a bidirectional association and it is a good way to illustrate associations.

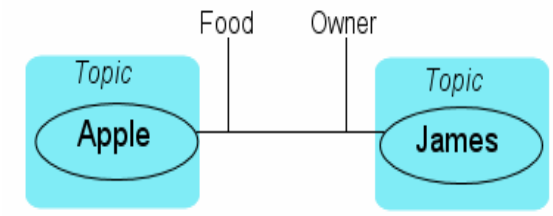


Figure 4 By assigning role types to the association, it gets more descriptive. "James is an owner of an apple", and "Apple is food for James". This can be red directly from the map.

Figure 5 shows how it is possible to add meaning to the application. This makes it easier to read the association.

From Figure 5 you cannot say that London lives in James. Topic Map developers solve this by scoping the "lives in" meaning to the person James. Then the application shows "lives in" for James and not when you are at the London topic.

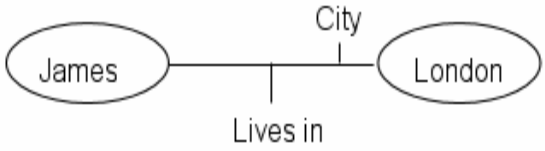


Figure 5 This is an association between James and London. A solution to how show the right way "Lives In" has scope is used.

Occurrence

A topic name, a topic occurrence, or a topic playing a role in an association is the available topic characteristics for a topic.

The XTM standard describes occurrence like this:
"An occurrence is any information that is specified as being relevant to a given subject. Occurrences constitute one of the three kinds of characteristic that can be assigned to a topic and are therefore governed by scope. Each individual occurrence is an instance of a single class of occurrence (also known as an occurrence type) that may or may not be indicated explicitly. The default occurrence type is defined by the "occurrence" published subject." – XTM standard.

As Figure 6 shows an apple can have an occurrence. An occurrence has a type colour and this type has a value green which is the occurrence data. Both occurrence data and occurrence type are topics. The Apple is a topic as well, and it is an instance of the topic fruit.

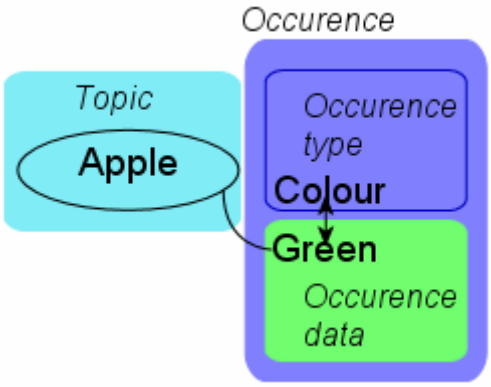


Figure 6 The apple can be described in many ways, and occurrence is one of them. Occurrence is attributes set to the apple.

Adding Information by Reification

Sometimes you need to give more information to an association. As mentioned above it can be confusing to give logical names to associations that are bidirectional. It is therefore possible to assign additional information to an association using reification.

Techquila [10] explains the purpose of reification like this:

"Of all the constructs in a Topic Map, only the topic is allowed to have names and occurrences and to play roles in associations. In other words, one can only make assertions about a subject which is represented by a topic. Those assertions themselves are not topics and so we cannot make assertions about assertions. Reification is the process by which a topic may be constructed to represent the assertion made by some other construct in the Topic Map. This process enables a name to be given to a particular occurrence of a topic, or documentation of an association to be "attached" to the association itself. "[2]

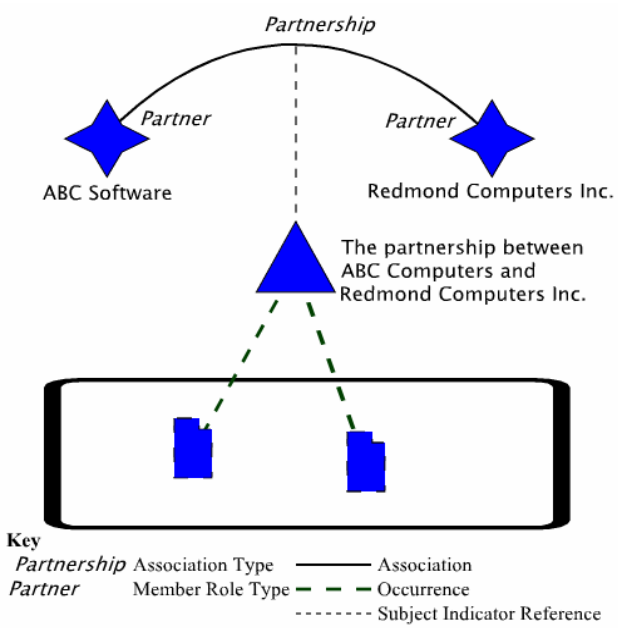


Figure 7 Example from Techquila illustrating how you can apply more information to an association

Reification makes an external topic that describes the additional information needed for the association. Figure 7 taken from Techquila, illustrates this with a partnership association between ABC Software and Redmond Computers Inc. When details of this partnership are added we use a topic to describe it.

Other concepts

The concepts above are the most important and are the ones we will use to model our ontology. For more information of the concepts above and other concepts you can see the Topic Map specification. An abstraction of this is available in Appendix B.

2.1.2 Topic Map Query Language

To perform searches is convenient to make queries. In relation databases this is done with Structured Query Language (SQL). In Topic Maps it is called Topic Map Query Language (TMQL). TMQL is not yet fully developed. But much research is done and people are working with it.

TMQL will make it easier to access information from the Topic Map. Today there exist Topic Map engines with an application protocol interface that developers must use to get and manipulate the information. By introducing a query language for Topic Maps the users can replace much of the implementation that is needed to get the information they need. Isotopicmap.org [3] has a central site on the Internet showing the ongoing TMQL projects.

[10] Techquila is a central site about Topic Map.

2.1.3 Topic Map Constraint Language

In models it is important to set constraints on how users are allowed to use and expand the model. The ontology contains the rules. The users cannot connect everything to everything when they are expanding the model. UML uses object constraint language (OCL) to specify the constraints. For instance a person can only have one age and a class can max have 30 pupils. It is the constraints that define these rules.

Topic Maps uses Topic Map Constraint Language (TMCL) to do the same. This is not a finished standard yet but several different projects are working with this. This chapter will explain what one should consider when setting constraints to a Topic Map. Isotopicmap.org has also ongoing a homepage [3] with evolving TMQL projects.

Setting constraints to the Topic Map

There are several issues to consider when you are setting constraints to a Topic Map. What kind of occurrences can a topic have? What associations can it have? How many instances can you make from a topic – if any? Which scopes are allowed to make?

The constraint defines what is allowed to add. This applies to occurrences, associations and scopes.

Topic – occurrence example

Recalling the apple example some of the topics are apple, James and colour. The occurrence type for the apple is colour. It is unusual to set James as an occurrence type. In larger Topic Maps it is extra important to consider this. The constraints must therefore state what occurrences which are allowed to add.

Association example

The associations describe the relationship between topics. Not all association should be made between topics. It is not natural to say that an apple eats James. The constraints must state what associations that are allowed to draw between topics.

Scope example

You can scope occurrences. An example of an occurrence is a description. A description can be given in different natural language – scoped to Norwegian, English or French. In this case it is not logical to scope the description occurrence to an apple. It does not make any sense. The constraints must state what scopes that are allowed to be added for topics.

Cardinality

We have now seen that it is important to set limits to what kind of information it is allowed to add for a topic. The concepts in the Topic Map can be seen as objects. The cardinality will be the same for topics, associations, scope and roletypes. We will now demonstrate different examples of important cardinalities.

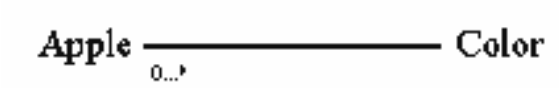


Figure 8 A color can be assigned to many apples.

Figure 9 James must have one, and only one age.

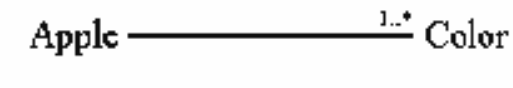


Figure 10 An apple has to have at least one color. Observe the inconsistency between the example in Figure 8. They would result in errors, and must be avoided.

Topic occurrence example

A topic can have several occurrences (0…*). A example of an occurrence for a topic apple is price. The apple price can of course vary from whether it is from you garden, or from a store. Another occurrence for the apple topic could be category, for instance country of origin. One of the terms for the apple could be that it has to be assigned to at least one category (1...*). The ontology can also state that apple has to have a description occurrence (1).

Association example

As we have seen one can assign role types to associations. It is not logical to make all associations between different topics. An apple has to be grown in one country but it can also be grown in several countries (1…*). The apple could be the ingredient in several cakes. Therefore the apple should be associated to several cookies (0…*). The ontology can also state that the apple have to have one creator association (1).

Scope

It is natural to scope occurrences for a topic. A description can be made in many different natural languages, like English or French. It is then usual to scope the description to the different natural languages. Also here you can use the cardinalities to set limitations to how many scoped you should add to an occurrence.

2.1.4 Modelling the Topic Map

The modeller can model the Topic Map in many different ways. As we have seen different concepts can be used to achieve the same goal. In this chapter we will give examples on how to make a topic and give characteristics to this. We will also see how we can use scope to give different views on things. After this we will explain how to use associations to connect topics to each other. We will also see how we can use reification to give more characteristics to the association. There is no “correct” method to model a Topic Map, but this chapter will introduce best practice of the different concepts. It is up to the modeller to use them.

Topic occurrences

The occurrence is often used to give further information about a topic. The occurrence differs somewhat from associations since it is connected directly to the topic. Occurrences are therefore useful for topics when you want to describe an attribute not associated to other topics.

Examples of occurrences for a person are age, height and weight. These properties are self explained and you do not have to use associations to explain that for instance a person is of age 19.

Topic types

Topic types are often used for describing the topics that are in the ontology.

Using scope

As mentioned above a scope is a certain view it is possible to have. You can apply it to occurrences, topics, base names or role types. In occurrences the scope is used to tell more about the instance. If a person makes a comment to an apple he ate, we might want to say who commented it. We can then scope the comment to the person. The apple can also have a description. This description can be available in many languages. We can make scopes for English, Norwegian and French and assign these to the description. The base names[11] can also be scoped to give different languages. The base name for apple is apple, but it can also be scoped to Norwegian and given the base name "Eple".

It can sometimes be convenient to scope topics. An example of this is under authorisation. An admin user should have more options then an ordinary user. We can then scope the users when needed to admin users.

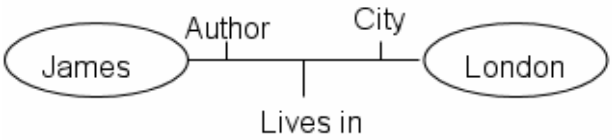
Association

The association is used when you want to describe how topics interact with eachother. You can then use the role types to specify how they are connected to each other. Typical example of this is when you say that a person lives in country and a country has the person as an inhabitant. Associations are often used since topics often interact to each other. Associations strengthen the Topic Map because it provides it with more semantic.

An association has no direction, but the role types can indicate some form of this. Using the scope feature it is possible to use a verb to the given association type for a specific topic. It is the Topic Map modellers that determine whether this is necessary or not. Using this one should be aware of association beeing bi-directional without indication of a direction. If the verb would confuse the person that reads the Topic Map model one should consider other solutions.

Lives in example

Some modellers are giving verbs to describe the associations. Since the association has no direction, the verbs can indicate that it has a direction. We will now illustrate this with an example Ontopia often refers to:



Figur 11 The "lives in" example is often used to illustrate the implications occurring when illustrating associations.

As we can see the author lives in the city. This is very logical to say, but the association has no direction. What if the association is turned the other way? "The city lives in the author". This is not logical. Since this association has two role types it is bidirectional. But describing the association with a verb indicates that it has the direction from person to city. It is up to the modeller and the intent of the model to decide how he will describe the association. It can be situation where it is smart to give verbs to the association and some situations it is not so smart. Steve Pepper a central person with Topic Map is giving his point of view here:
"So should you avoid verbs? I think not, provided you don't imply a directionality that is not there. I use them quite freely as the IDs of topics that type associations. However, I also use multiple names for such topics and tell my application to choose the most pertinent name based on the vantage point from which the association is viewed" - http://www.infoloom.com/pipermail/topicmapmail/2002q1/003610.html.

Reification

As Figure 7 shows; the example is about reification and it is used to give more details. The example is about an association but this could also be about a topic or an occurrence.

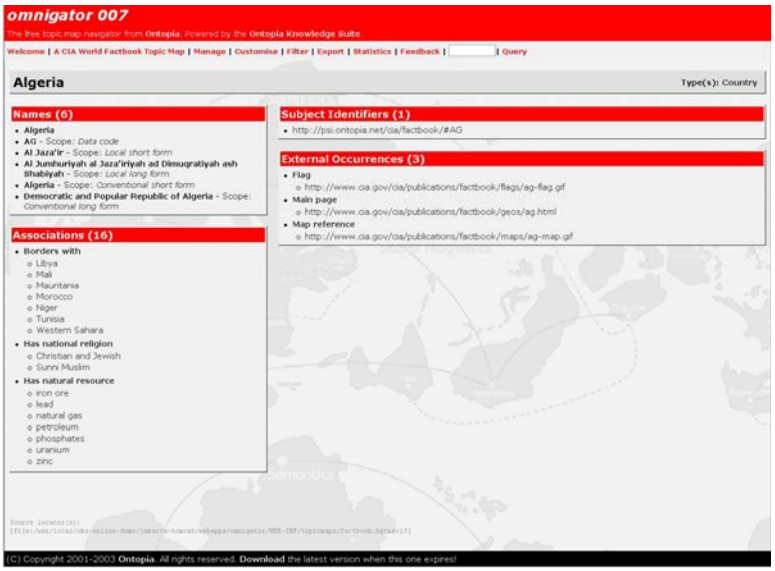
[11] Base name – The name a topic must have, a topic can also have multiple names.[A]

2.2 Visualisation techniques

Several visualising techniques can be used to represent and navigate in the knowledge layer. This section will describe the different techniques and see how it is connected with the semantic provided from Topic Maps.

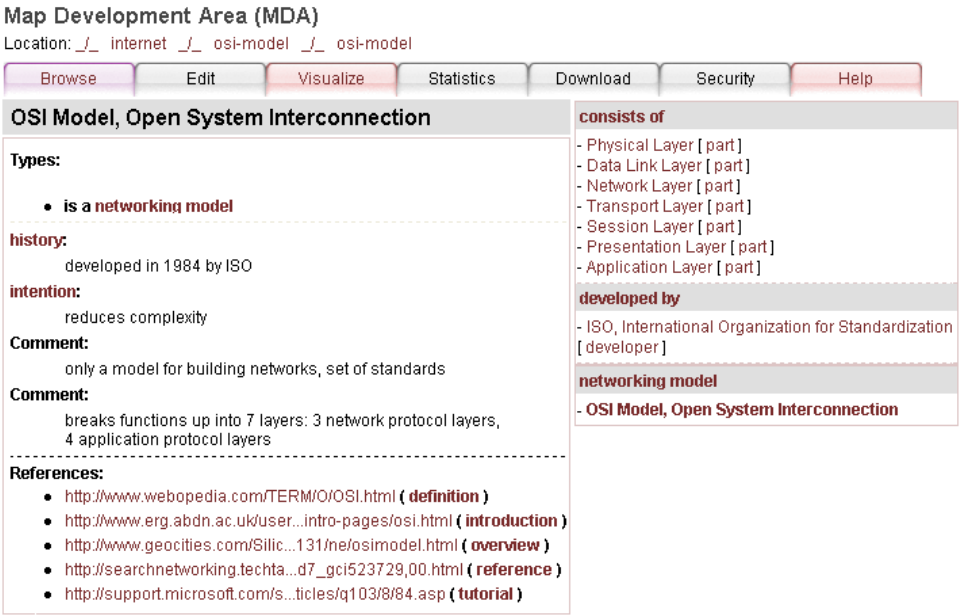
2.2.1 Text-based

The most common way to represent information is with text. Fonts and styles applied to the text structures the information and set focus to the important text. Illustrations will also help describing the text. It is common to make menus where you can navigate.



Figur 13 Omnigator has this view, and it is possible to explore it more on http://www.ontopia.com/omnigator. Omnigator will be investigated more with more pictures in the review part of the thesis

Topic Map engines often display lists or indexes, so the user can view the related information for a topic. The Ontopia navigator, Omnigator, is a typical example of this. The Omnigator lists all the information of a topic and by using hyperlinks you can click on an association or a topic and get the information about this. Figur 13 illustrates an example of this visualization. The design in Figur 12 is used in Bond University. Roberta Barta and Anitta Altenburger have developed the site. Student's use it to present different internet related topics in either a text based mode, or a graphical mode using tree structure.



Figur 12 Bond university use a text presentation mode, and a graphical presentation mode to visualize maps made by students in projects. The graphical view is presented below in Figur 17. They have managed to create a nice design, with a nice layout

2.2.2 Tree structure

It is also more and more common to present information using a tree structure. A more graphical approach is then reached. Information is often expressed with elements like bubbles, boxes or some other sort of figures. The elements are connected. Graphical interfaces like these might provide a clearer overview so a user can easier understand how the information is connected together. Figure 15 shows a family hierarchy. You can see that John is parent of Chris and Karen, and Karen is parent of Jane and Tim. The tree structure gives a good overview of the family. Genealogists often use this presentation to show many generations of relations in a family.

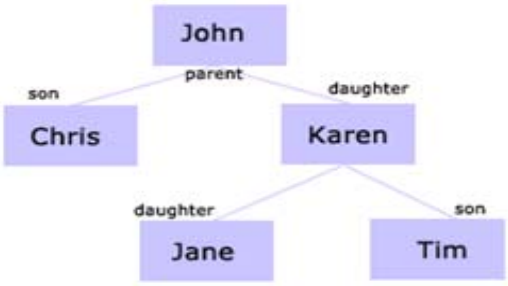


Figure 15 The tree structure, here used to visualize a family hierarchy.

2.2.3 Graph

The graph is an extension of the tree structure. Tree structure has a starting point that the other nodes derives from. The graph structure is much more scalable, because you can give many associations to the node. We insert a new item to the previous family chart, IBM, and John and Tim works for them. It is now a graph structure.

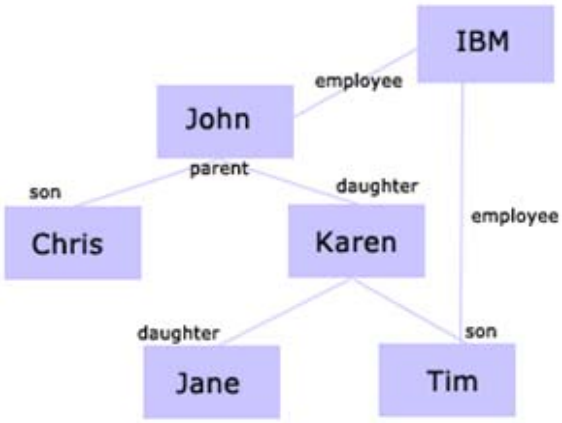
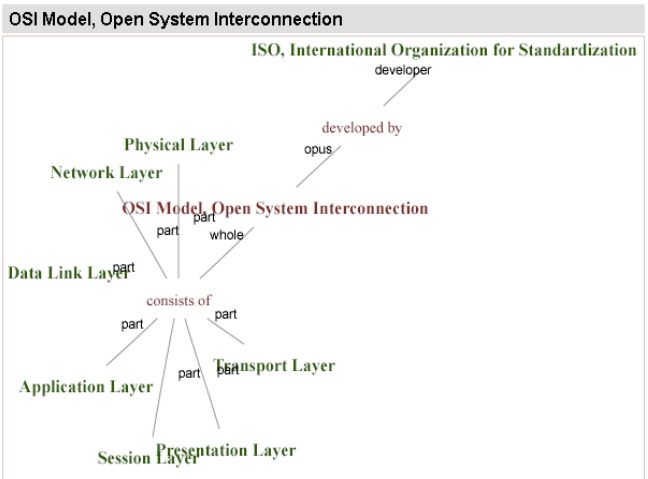


Figure 14 The extension of the tree graph with the employee association included. The tree graph could not support associations like this.



Figur 17 Bond University uses scalable vector graphic to present Topic Maps in a view assembling a tree graph (star schema). Robert Barta send us in request the simple script she uses to generate the view. Adding more intelligence to the generation of the map would possible avoid overlapping texts.

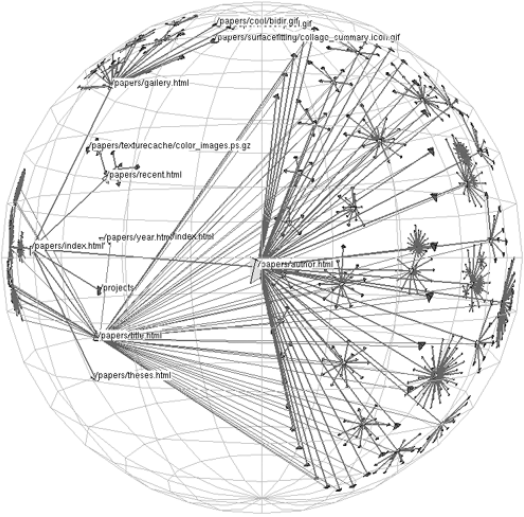
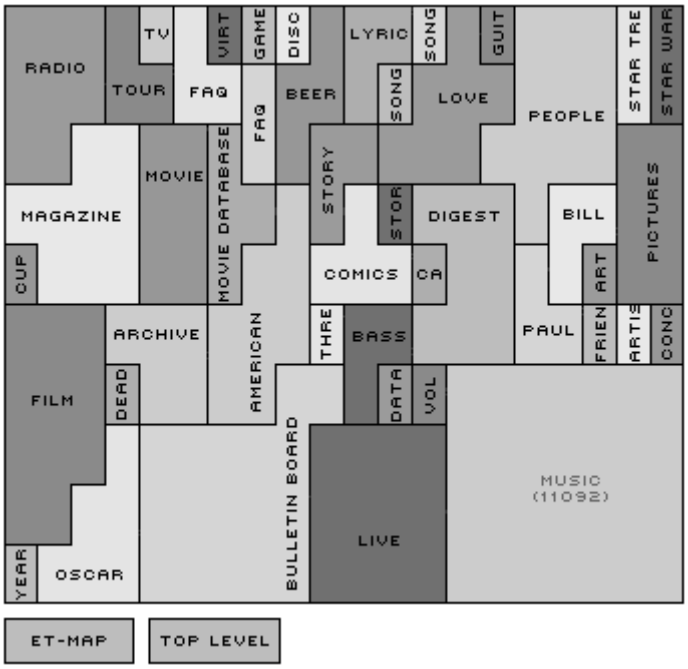


Figure 16 This rather unconventional way of exploring Topic Maps is used in the German thesis “Applications of Topic Maps for easy navigation in digital documents”- Ronald Heckel. Investigating it closer reveals that it is tree graph visualised on a three dimentional globe.

2.2.4 Map

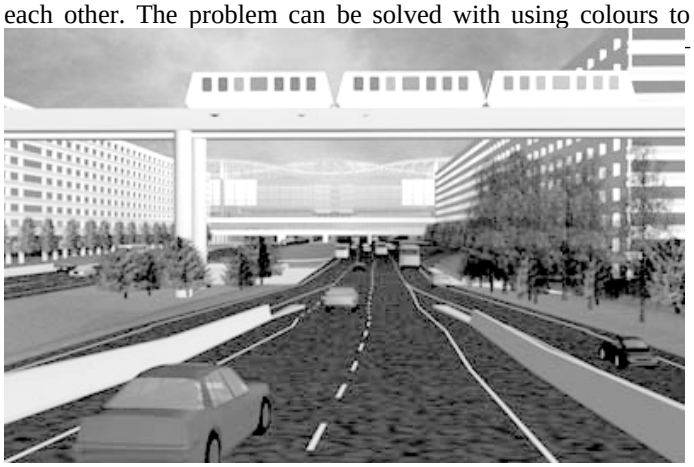
Map is another graphical approach. This kind of map is not as common as the other techniques mentioned above. It is perhaps not so easy to understand at first. An advantage of this map is that you can add very much information into an area. Figur 18 from The Topic Map[5] shows one way of drawing a map. Information relevant to eachother is automatically neighboured. A magazine is connected to movie, cup, film, radio and tour. The map in the figure has 3 dimensions. Colours of the blocks can be used to describe associations. The word it self says that it is a map with topics, so structuring Topic Map with maps can be very useful.



Figur 18 The map is not very much used. Click on the term you want to investigate. The map is not appropriate if there are many associations with elements with a long distance between each other. The problem can be solved with using colours to

2.2.5 3D Visualising

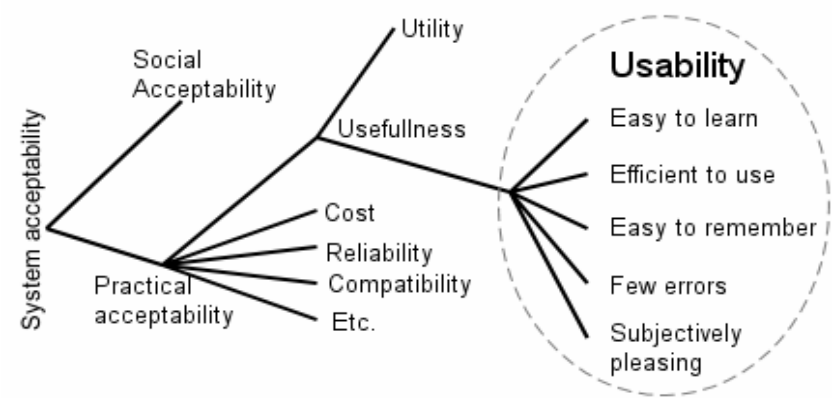
By introducing a third dimension it will be possible to present relations and descriptions in other ways. However it can also become more confusing for the user. It is therefore important to give a well defined structure that people understand. People does research on different methods to present data, and it might be more common in the future to use 3d. (Picture taken from planet 9 studios, Inc San Francisco)



Figur 19 3D visualisation is close to the reality and many people can deal with this visualization when it gets easier to use this approach both in bandwidth and usability of the application. This picture is from XML topic maps [F]

2.3 Usability

Usability is traditionally associated and defined with five attributes: Learn ability, efficiency, memorability, errors and satisfaction [6]. There are also other broader definitions as acceptability. Acceptability (Figur 20) includes the user, the clients of the user and also the managers. Practically all persons involved some way with the application. As the figure shows there are many aspects regarding applications and their environment. We will consider usability for the end user of an application, and shortly comment the utility factor. An application has high utility when it solves a task in a good way. One example is a car. It is made to be driving with, and it solves that task. A car has high utility when it comes to drive from one place to another. One can do more research on acceptability and other factors by entering Jakob Nielsen homepage at www.useit.com . This webpage along with other literature Jakob Nielsen has written is the main resources we have used. He is regarded as one of the people with most experience in the profession of usability. Jakob Nielsen is also friendly referred to as “The king of usability” [5] and “the world's leading expert on user-friendly design"[6] among other marks of respect. Usability also applies to other aspects as installation and maintenance. We will disregard these because maintenance and installation is not responsibility of our target user group. In this chapter we will explain the five usability attributes and why these are important in order to make a system user-friendly. Several criteria should be applied to an application in the developing phase. The book Usability Engineering by Jacob Nilsen states that: “It is important to realize that usability is not a single, one-dimensional property of a user interface. Another important issue is to check how good the usability is for your system, and we will discuss this afterwards.



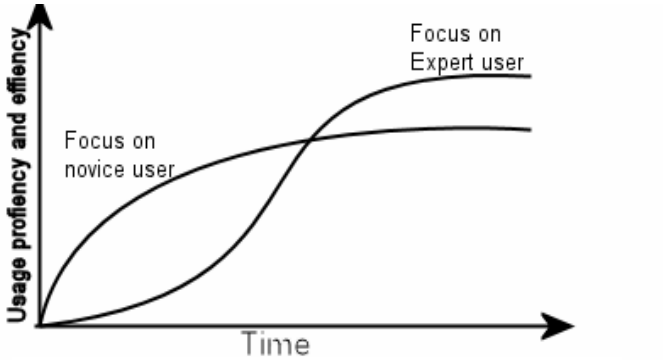
Figur 20 A model of the attributes of system acceptability (Jakob Nielsen) The themes inside the dotted circle are narrowed down to be called Usability. Usage of these themes and how we will use them will be explained in this chapter

2.3.1 Learn ability

Learn ability is the first attribute you meet when you are introduced to an application. All users must learn the application either before, or while he starts using the application.

Learn ability curves

The Learn ability attribute can be measured in learning curves. What kind of learning curve it has, depends on what kind of user the application is intended for. Figur 21 shows two different users with typical learning curves. An application intended for an expert user often takes longer time to master, but the user will reach a higher profiency and efficiency level. The novice user will quickly learn to use the application, and he will not reach the efficiency an expert user would on an application designed for that particular user.



Figur 21 Depending on who the system is built for gives different learning curves. The Expert user ends up being a faster and more efficient user. (Picture from Jakob Nielsen: Usability engineering)

Almost no users manage to use all the features an application offers. They tend to learn what they need in order to use the program. After reaching a certain level they often stop learning.

Techniques of measuring Learn ability

Learn ability is one of the easiest tasks to measure. This is because you can assign a task to the user, and measure how long time it takes before he masters the task. After the task is completed you assume they have learned it.

You could also test learn ability by assigning a specific number of predefined tasks and say the user has learned the tasks at the moment he can finish it in a given time. Errors made by the user should be avoided by the application. When errors can not be avoided the errors also could be used to measure when the user has reached a satisfying level of profiency.

2.3.2 Efficiency

Efficiency is referred to the state where the curve in Figur 21 levels out. Some time has gone, and the user has reached a level of efficiency, and profiency using his application. The time a user use to reach this level differs for each user and what kind of application it is. According to researches users tends to level out as soon as they have learned what they think is “enough” [6]

Techniques of measuring efficiency

When the learning curve flattens out (Figur 21) it is time to see how efficient the application is. To perform this measurement you need experienced users. This is because he must have undergone training, or he must have some other experience with the application. To select persons to do this measurement, you could for instance define a user with one month, or one year of experience with the application. To get an even better measurement, you could measure more than one user. After selecting a group of persons, you must define a set of tasks they must finish. The average time they use is then defined as the efficiency. You could also set a user to do a specified set of tasks continuously. He will then learn how to use the application more and more effectively. When the time to do the defined set of tasks levels out, as the learning curve flattens out in Figur 21, the efficiency of the program is reached.

2.3.3 Memorability

To explain memorability we want to introduce a user called the casual user. The casual user belongs to the third important user group in addition to the expert and the novice user. Jacob Nielsen defines the casual user this way: "Casual users are people who are using a system intermittently rather than having the fairly frequent use assumed for expert users." These users have used the application before, and therefore they have some knowledge about how it works. Therefore they do not need to learn the application again. To the casual user the application must provide easy to remembered functions from time to time. Ordinary applications used by expert users and novice users also demand some memorability. If an worker goes for an holiday, or is absent of other reasons, he should be able to continue his work rather fast after returning. In general interfaces with high memorability is wanted, and it is often said that improving learn ability also improves memorability [6].

Techniques of measuring Memorability

Memorability is not that often tested thoroughly. When you do it there are mainly two ways of doing it. The first one is as simple as testing a user that has been absent from the application for a while, and test how fast he can complete a set of tasks. The second method is to test a user's ability to remember functions, pictures, icons, and other important issues from the application.

2.3.4 Errors

In general errors made by the user should be avoided. An error is an unwanted result of any action done in the application. Count the errors over time, and you get the error rate. Errors in general are not wanted. There are different kinds of errors. Some errors are catastrophic, and some are just minor errors made by the user when he for instance types a wrong character. Minor errors can often be corrected at once by the user. Catastrophic errors ought to be impossible to do, or at least it should be difficult to do them. Undo and redo actions are then good solutions to implement, especially when user interaction can result in catastrophic errors.

Techniques of measuring errors

Errors can be counted when they occur using the application. There must be a difference in catastrophic errors and minor errors.

2.3.5 Satisfaction

The satisfaction is a subjective feeling the user has upon an application. It is often referred as to how pleasant the system is to use [6]. Pleasant is a wide term. It depends on what the user wants to do, and what the utility of the application is. Game industry on Internet wants people to enjoy spending hours and hours in front of the screen playing. For the internet gaming industry satisfaction is extremely important, their existence depends on the satisfaction of the users.

Techniques of measuring Satisfaction

Different techniques of measuring satisfaction exist. Physical tests like EKG, heart rate, pupil dilation, skin conductivity and blood pressure are used [6]. Ordinary tests asking for the users opinion is also used. The average answer from multiple users should be used when asking more than one user of his opinions. The average will rule out some of the subjectivity introduced by the users. It is important to let the users test the applications thoroughly before answering the surveys.

3 The InterMedium case

3.1 Background information

Collecting information from resources on the Internet and presenting them to their customers is the main focus of InterMedium. Customers have questions they want to get answered, and they have different needs. These questions form the base of specialized harvesting from, and searching after different Internet resources as newspapers, news agencies, competitors, WebPages and other resources of interest. The result of these searches can enlighten and answer questions asked from InterMedium customers. Decision makers get pre-sorted information based on automated intelligence from InterMedium. A larger foundation including "outside the firm" information can then be used when important decisions are made. One example of outside the firm information can be the public's opinion about one product. This is also often called "Competitive Intelligence". InterMedium is a full service provider of "Competitive Intelligence" and they describe it like this:

"The activity of monitoring and analysing the environment external to the firm for information that is relevant for the decision-making process of the company" – *www.intermedium.no*
Intermedium is a member of "Society of Competitive Intelligence Professionals"(SCIP) and SCIP describe Competitive Intelligence as :

"A systematic and ethical program for gathering, analyzing, and managing external information that can affect your company's plans, decisions, and operations." -http://www.scip.org/ci/

Technique

Harvesting of all these different resources implies a well structured knowledge layer in the background. Knowing that an impressive amount of 20.000 articles per day are piped through the InterMedium system emphasises the need for a solid knowledge layer to "increase information quality and relevance" and "Decrease information redundancy and noise" as stated on InterMedium own information page. One case we have got from InterMedium focuses on the new "Word of Mouth" concept. They also want us to consider the possibility to generalise the concepts to other ontologies.

Word of Mouth

"Word of Mouth" (WOM) is as simple as people having opinions and discussing these in one or more subjects. WOM is general talk or opinion of maybe uncertain reliability and relates into words rumours and reputation. An InterMedium customer wants to monitor WOM regarding one or more issues. One example could be a company wanting to monitor people's feelings about a new advertisement campaign running on TV. Another example is a company selling a product, and they want to know what people generally have to say about this product. It boils down to discussions and opinions about something an InterMedium customer is interested in. InterMedium has to find, sort out, save, and present this information.

3.2 Describing the InterMedium ontologies

3.2.1 The Word of Mouth ontology

An ontology made in cooperation with InterMedium will be the foundation used to structure WOM information. This ontology (Figure 22) is used as a demonstrator, and is a somewhat simplified, and easier to understand, version of the real WOM ontology.

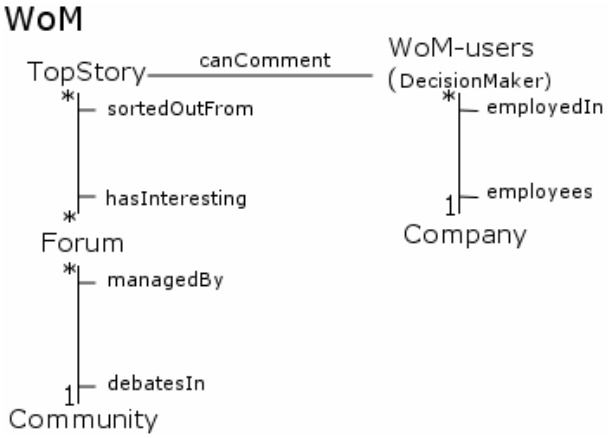


Figure 22: WoM – Word of Mouth. The InterMedium Case with role types, and cardinality. This is a simplified version. Attributes are for now not shown in this ontology

WoM-Users, and Company

InterMedium wants to answer its customer's questions and needs, with help of this model. WoM of interest can then be extracted and commented. These customers are companies' with decision makers and company strategy developers, here called WoM-Users. These users will both extract knowledge from the WoM, and add information, for instance comments. InterMedium's own employees will also act as knowledge adders. They will also add or edit information of interest to enhance the knowledge embedded in the WoM.

TopStory

When a company wants to monitor what people have to say about for instance its release of a new vacuum cleaner, the foundation for a story is made. Different attributes will then be used to monitor what kind of level this story has. Such attributes determining whether a story is a TopStory or not, could be the number of participants in a debate. InterMedium has decided that top stories are important and that they should be available for the WoM-users. Some stories have varying participation by its participants from time to time. Some stories ends completely after a while and some begin hibernating just to become active after some time. TopStory is the way InterMedium describes a specific topic of interest that has attributes which can make it interesting to be observed by the WoM-Users. A TopStory can be debated in many forums, not only in a single thread in a forum.

Forum

A forum is the place where debating appears. InterMedium has a giant list of forums they search through everyday monitoring the activity there.

Community

A Community consists of managers, and all other users of the forums.

Sorting algorithms

Sorting algorithms to determine whether a story is top, or not, is provided by InterMedium. These are not essential to our task, and we will only use test data herein.

3.2.2 Other Ontologies

Another ontology (Figure 23)will be used to show the concept of proof. This is a simple ontology, using other role types, and other topics than the WoM. The ontology consists of information sources, news within, and company with interest of reading the news. News will be simulated as if they were sorted out by InterMedium web agents.

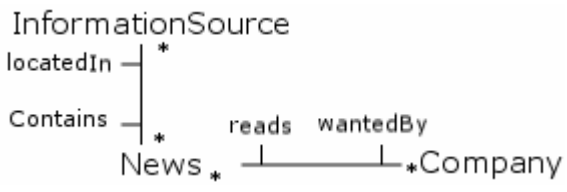


Figure 23: The News ontology. In this ontology companies wants to read news from different sources regarding what they has defined as important.

4 Review of existing visualization applications

The reviews are made to measure and compare existing model driven applications. The goal is to learn from and if possible build on already existing work. The reviews depend on the utility the application and its user group. In the following chapter we will define the user group, explain the usage of the application and give criteria that are important to help us later when describing an ideal solution.

The user group

The user group is the persons that will use the application. Some products evaluated can be used by everyone while other can only be used by educated users. There are many different users with their own perceptions. Our user group will consist mainly of users without prior knowledge of the backend technologies. The reviews will also mention expert users if needed. Model driven applications based on Topic Maps are not fully developed. A relatively high level of education is needed to use this specific technology. A novice user will sometimes have difficulties using some of the applications because he has no or little knowledge in Topic Map concepts. The reviewed program can still have some good qualities for an expert user. This will avoid hiding some of the qualities in the applications unavailable for a novice user. These qualities may be used and rebuild to benefit the novice user in an ideal product.

The usage of the application

Describing the usage of the application will help to define how the user interacts with it. The user using the application should have possibilities to navigate and edit a model based on ontology. Ideally he could import the case from InterMedium and start navigating and editing this. If the program lacks possibilities to import and work on the case, we will use the available ontology's or models in the program and make a review of the concepts for this application. These concepts can later be used to describe the ideal product.

Using the InterMedium case in the application

Working with the chapter 3 (InterMedium Case) the user should browse the communities and view details of the associated forums. Top stories are especially interesting for the user to comment. They should also have the possibility to insert and change information from the ontology.

Evaluating existing applications

One must consider many criteria evaluating different existing applications. A criterion has to be well defined. There must be as little as possible or no misinterpretation so everyone understands it correctly. A list of criteria are presented and explained thoroughly below. General perception will differ from a novice to an experienced user. The reviewing criteria will, when necessary, consist of separated parts; the experienced, and the novice user. Using these users can give valuable extra information. It will clarify the criteria since these two distinct users sometimes have completely different views. A 5 point scale (se the reviews in appendix A, and chapter 4) will be used to reflect the application's solution to the criteria. Applications with very good approaches to one task will be rewarded.

4.1 Review Criteria

Good evaluations are difficult to make since they should be used by a group of people, with different preferences. The skills involved by test persons are essential for test results. The programs we will evaluate are not mature and the technologies involved are unfamiliar to most people. It is therefore difficult to involve real people in this test. We will try to make an objective evaluation on how a novice user and expert user will perceive the criteria.

4.1.1 Usability

As discussed in chapter 2.3, usability is important for the overall impression the users gets for of an application. The program can have features but they are irrelevant if no one understands these or if they are difficult to use. As discussed earlier usability has five attributes (chapter 2.3). These attributes are somehow involved in the criteria for evaluating the applications. The focus is on the learn ability and efficiency. The other usability factors differ for the applications. They will therefore be considered when they affect the program. These attributes are considered in all the other criteria, since they are indirectly part of them. They will however be evaluated as a separate part. the Following chapter explains what will be taken into consideration when evaluating the usability criteria.

Learn ability

A high level of intuitiveness increases the ability to learn a program. An intuitive overview assembles a drifting feeling giving you a natural progression when working with it. All options should give valuable and insightful information. Another important consideration is the entry level. The entry level indicates how much you need to know before starting using the application. Do you have to consult a manual, or even attend a course, before starting to use this program? Both the entry level and the intuitiveness will differ from the skills the users have. It is also important to measure the time a user spends to achieve a sufficient level of proficiency to do some actual work. We have no possibility to measure how much time a user spends to learn the program and will disregard it in the evaluation.

Efficiency

The efficiency of a program depends on what the program is intended to do. The programs we evaluate are mostly of the same kind with the same purpose. It is interesting to see if some of them are more efficient to work with compared to another. An efficient application will both satisfy the user, and the potential employer paying the user.

Other usability factors

The remaining usability factors are memorability, errors and satisfaction. These are explained in the end of chapter 2.3. These will when necessary be commented in the written part of the review.

4.1.2 Navigation

Navigation is what facilitates movement from one part of an application to another. There are clearly advantages when this navigation is intuitive. The understanding of a given application is often based on previous experience. One must consider questions like: Does it resemble what it is supposed to give a picture of? Is there anything you have to know about the navigation, which is impossible to find without a manual? The following questions will be used to measure the navigation of the different applications we will review.

Browse

The view used when manoeuvring the application gives you certain options. Features like the “back” and forward button could provide a better and faster navigation.

Browsing is what you do when navigating around in your model. The supplied options must improve the user’s capability to browse. It should enhance the navigation.

Searching capabilities

Even the best designed system can be challenging when it gains complexity. Searching capabilities of some sort can help finding what is wanted. A search can be done in an alphabetical ordered list, which has linking functionality to the information layer. Search engines where the user can enter keywords can also be a good help. The latter can provide better results when complexity is increased. A dictionary is a good example of an alphabetical ordered list, and Google[12] as a search engine. We have chosen to compare these as one criteria since it often will be tailored which one to use, if not both.
Important questions considering the searching capabilities are: Does a layout exist, or a scripting possibility providing search functionality? Is the functionality easy to understand?

Model overview

A model overview in our case is actually a map over the model. The application is intended to show a model, but the model can be large and incomprehensible. The model overview can give a faster impression of what the map can offer, and even where to find it, in the map. The criteria also consider how the map is presented in order to get the overall picture of the model.

4.1.3 Edit

Information is usually stored in databases and files. Generally people do not know how to access lower level mediums and should most often not be allowed to either. It is important to set limitations to what a user can edit. The user should therefore only be able to edit the model or system where he is allowed.

Insert new instances

A model has a framework giving the ontology. Instances from this ontology are what users can add. Considering the InterMedium case it should be possible to make comments and insert and edit communities with associated forums and more details. All these are instances from the ontology. How easy is it to insert new instances? Some programs have scripting systems, and it might be more suitable than a simple drag and drop based editing system for an experienced user. The novice user would probably need some education to perform advanced handling like this.

The user should insert instances with his premises. Do the application lay unnecessary boundaries on the user? A good program for the user is if he easily can insert information that he is allowed to do without a hassle.

Modify instances

Modify an instance is one of the actions which is done often. Modifying must be implemented if a user wants to change something. With this criterion one must consider how easy it is to modify an instance, and when you modify anything, it should be unambiguous and clear what you change.

Modifying instances and information generally should be easy and faultless.

Delete instances

Instance expired or inserted with a failure should be possible to delete. This should be available in an understanding way. It should not be possible to do catastrophic errors in an application, and it is especially important under this criterion.

[12]Google – The most popular search engine on the Internet in 2004

4.1.4 Import/Export

Many programs today are specialised to one specific area. Some are more suited for browsing and some for editing. To combine programs it is necessary to export the model to a format supported by other programs. This review focuses on Topic Maps. Therefore the import/export criterion must consider to what extend it is possible to import and export the XTM format. Other standards preferable based on XML, will also be considered since it is possible to go from one XML format to another.

4.1.5 Design and Layout

Design is often referred to as the drawing of an idea, and layout is the placement and order of the elements herein. The first impression a user gets of an application is often connected to the design and layout.

Reviewing the selected model driven applications in this thesis will consider:
Is relevant information visible and illustrated in a professional and structured way? Has the design features (text, figures) to present the model? Does the core information get your attention? Do the buttons give any sense?

Presentation of a model

A model can be presented with text, figures or a combination of these. Evaluating this criterion will help us to see when text is preferred before figures. We will consider where it is appropriate to use figures and were it is appropriate to use text.

Illustrations

There are many icons, pictures or figures that can illustrate the purpose of an object is intended to do. Do these illustrations describe the meaning itself? Are the illustrations so incorporated and commonly used that people know what it is for and what they mean?

Available core information

Every program has its own speciality whether it is for playing to music, or for reading news. Available core information about each applications speciality should be visible. It is important for a program to have this information available.

4.1.6 Application specific

The criteria mentioned here is not part of the evaluation directly, but they will be included because they have some aspects that can give important information. These criteria will be commented if necessary in the reviews. The following keywords are used in this part of the review: stability, technical quality, robustness and maturity.
And the Following questions are considered:
What developing environment has been used to create the application? Are the libraries available for use today? Is it possible to implement components from on an ongoing project? Are the last releases of the application stable releases? Is the application general, and scalable? Has the developers announced new improvements in foreseeable future? Is the application in an early stage of development? Is the application mature?

4.1.7 Criteria table

Criteria	Very Good	Good	Average	Poor	Lacks
Usability					
Learn ability					
Efficiency					
Navigation					
Browse					
Searching capabilities					
Model overview					
Edit					
Insert new instances					
Modify instances					
Delete instances					
Import Export					
Import ontology					
Export ontology					
Design/Layout					
Representation of a model					
Illustrations					
Available core information					

This table is used throughout the review. It can be found in appendix A, and the results are written in the review summary chapter following.

4.2 Selecting applications to review

We selected applications and used a couple of rules during the selection. The applications had to use technology similar to or equivalent as Topic Map technology. The Brain, Omnigator, TMNav, TMtab, Naked Objects are applications we chose to evaluate with the predefined criterions. Thorough descriptions and analysis of all applications is carried out in the review chapter. We choose these applications because they were similar, and at the same time unequal. Different techniques in the applications was used to browse a Topic Map, or similar. The applications we did not review was either to close in similarity to the programs already in the list, or they had other issues disqualifying theme.

The Bond University Topic Map presenter [8] is developed by Roberta Barta and content is build by students and affiliates. We did not choose it is because of its similarity to TMNav in the graphical mode, and the similarity to Omnigator in Text mode. Bond University was better than TMNav and Omnigator, and we have observed the qualities of the design and layout in their applications.

Location of the Reviews

Thorough reviews of all applications can be found in the appendix. All criteria mentioned in this chapter are applied to the applications to help us in the rest of this thesis.

4.3 Review summary

We have now investigated some applications. The most important concepts of interest that appeared during the reviews will be written here. The same structure we used during the reviews and when defining the criteria's will be used. We will only consider the essentials from the reviews, and you can read the full reviews in the chapter. All scores are based on our personal experience with the applications, and we have tried to be as objective as possible.

4.3.1 Usability

Learn ability

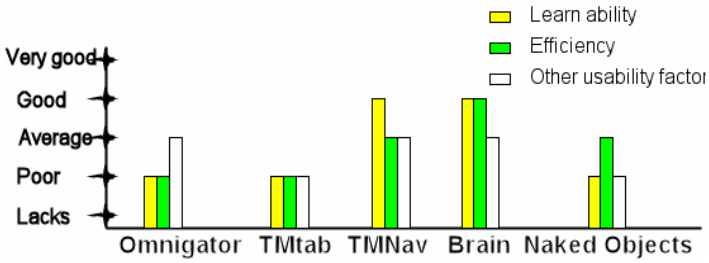


Figure 24: These important usability aspects are based on Jakob Nielsen research, and he's book Usability engineering.

Brain and TMNav got the best score on this criterion. Simplicity of both programs was a key factor. Simplicity gives often better usability because of lesser options to concern with [6]. TMNav had very few functions, but those provided was at all times visible to the user. The Brain had many functions, but still the developers of the application managed to provide these functions in a way making them feel easy to use. A program with easy and understandable functions will quickly give you a start with extracting information from the model. In addition Brain had a very well built manual, so when something was not clear it only took a few seconds to find the issue in the manual. The best thing would be an application without the need of a manual [6], but it is ok to have the manual there because people interpret functions differently. TMNav did not have a good manual, and it was really not necessary. TMNav is highly specialized to browse Topic Maps, and therefore the few functions were Topic Map related. To understand TMNav you could study the XTM specification, something people in general will do not. Without studying the XTM specification, you still got most of the information you needed from the Topic Map. The entry level in both programs was low, TMNav because the only thing you had to do to start was to load a map, and Brain because of the very easy to use wizard giving you a quick start.

Efficiency

Efficiency was defined as how fast a user can use the application. If the programs were very similar we could take some real tests with timing, for instance measuring how fast it was possible to insert five different things into the application, or how fast it was possible to find information. We regarded this as unnecessary because most of the applications had in fact very low efficiency, except Brain. Naked objects had some interesting thoughts about efficiency saying a user that had much customisation options and much interaction with its application would be a more indirect efficient user than one with a highly specialised program only considering fast use. The idea seemed good, but the implementation in Naked Objects missed a bit, maybe because they used it all the time as a proof of concept in the demo. Also tests have shown that novice users, typically like our user groups, most often will not use customization options [6]. Brain got better score on efficiency because of the easiness in most functions implemented. The search engines in Brain were also very good, and integrated in a very good way. When pressing keys, Brain automatically assumed it was a search. This brought up topics starting with the letters you pressed. Features like this were very nice if it was a long way to manually browse to the topic of interest, the speed of using the application was greatly enhanced.

Other usability factors

Naked Objects have a high priority on usability factors, and they are still developing their product. Still they have scored somewhat low on this criteria. They have some good ideas, but other elements shadows for the good ideas, and degrade the score. Common features most users are familiar with from Microsoft Windows have been neglected and they have invented new ones. This is only one of a few such things. They will probably evolve into something better as time goes by because they have some very good ideas, but for now it is not good enough. Brain and TMNav scored best on this criterion, again because of the simplicity involved in the functionality provided. You can return to the programs after a long time and relatively fast start using the applications again. Their views are nice to look at and you have a positive feeling when using them.

4.3.2 Navigation

Browse

Brain has once again got a high score together with TMNav because of the simplicity. The browsing features in Brain are good, and it is easy to navigate from one part of your map to another. TMNav is also good to browse in, but if there are many topics, they do not have such a good way of showing them as in Brain. TMNav has a random function with some intelligence to avoid writing two topics at the same place, but sometimes it does not work, and you have to redraw or modify the map. Brain draws the topics in a structure, instead of drawing it randomly, and it worked better than the random function in TMNav. When there were too many topics to draw up, Brain used a good solution consisting of a scroll window to visualise all topics.

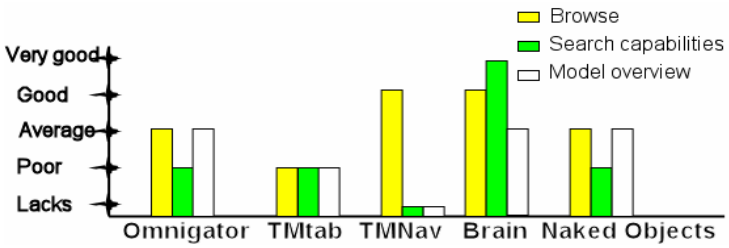


Figure 25: Browsing, searching and model overview are important aspects an application provides to the user.

Searching capabilities

Most of the applications scored low here. The search methods were either to difficult to use, or they lacked some features one usually need to do a search. Brain did again get a good score, and they have implemented three different search engines which in fact did not feel like a crowd. The simplest one was only a topic finder. The two other search engines were more specialised, giving you the possibility to search after files, strings, or any other information you could think of. Omnigator had one simple search function, and the possibility to use Tolog[13], a powerful query language, though such features cannot be used by novice users.

Model overview

All programs had an average score on the model overview. Omnigator and Naked Objects were the only programs offering something close to an overview. Naked objects start out with showing all possible elements you can add, and Omnigator had one optional auto generated overview of the Topic Maps. One reason of the rather average overviews can be the fact that it is relatively hard to make a reasonable model overview automatically. One solution to this could be to tailor the overview by your self.

[13] Tolog – Topic Map Query Language [12]

4.3.3 Edit

Insert new instances

All programs other than Brain and Naked Object scored low in this criterion. The other programs scored low mainly because they are not created for ordinary users. Brain is a commercial product and they have to consider the users. Naked Objects are developed with usability in mind, so it was natural that they scored well in this criterion. Brain used a drag technique and Naked Objects also used this technique, in addition to an ordinary double click to create a new instance. Brain got better score than Naked Objects because of some errors appearing in the Naked Objects implementation.

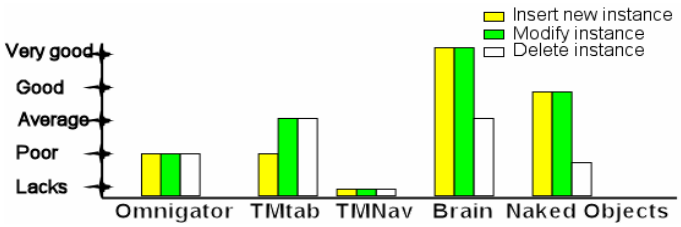


Figure 26: One should think insertion, modifying, and deleting of instances was almost the same for all applications with no complications, but the reviews showed us some differences. The reviews in the appendix has more information then presented here.

Modify instances

Modifying was easy in Brain and Naked objects. Brain had the best score again, because everything you could modify was available at all times, and it was easy to understand how to change and modify. Changing an association in Brain was made in a very nice and understandable way.

Delete instances

Deleting was relatively easy in all programs, but all lacked undo options, except Brain. The delete functions should not be close to another function you often use, especially not if you have no undo possibilities. Naked Objects had this problem. By designing the application this way, you will surely invite errors mistakenly done by users.

Import/Export

Import and export options were mostly provided in a well arranged way in the applications with this function. TMtab lacked options for importing existing Topic Maps in XTM format. We selected not to rate the import export functionality because when they was provided, most of them was easy to use.

4.3.4 Design and Layout

Representation of a model

Many different ways of presenting a model were used in the different applications. TMNav uses an open source library called panckoucke to draw the map, and it did a good job with visualising the model. Brain used its own a bit futuristic way of showing the model.

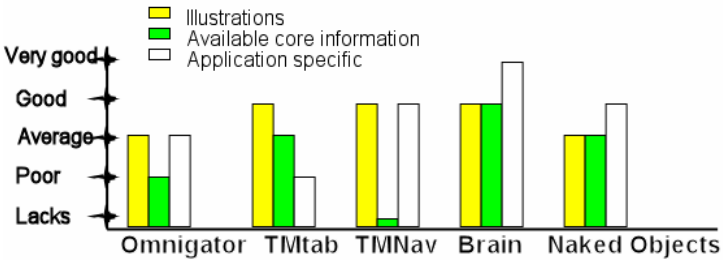


Figure 27: Illustrations, available core information, and the application specific criterion are the last review criterions we used to emphasize differences in the applications.

Illustrations

Naked objects used a form of workbench, and you created instances with a figure of what that instance were supposed to do. The icon to show what kind of object you used gave much help. Brain had no drawings directly in the model, but they used a rotating graphic (see picture in Brain review in Appendix A) to emphasise which topic was in focus. You could also assign a picture to a topic if you wanted to, but this enabled it for all topics, and it should be possible to assign it to only those topics you wanted to.

Available core information

TMNav, Brain and Naked Objects had the best score here because they were more directed to users who are going to extract information from a map. TMtab and Omnigator are directed against developers of Topic Maps, and therefore the functions provided there are also more headed towards such users.

Application specific

The application specific criterion had many issues, and these could help us when choosing what kind of libraries we wanted to use when developing the prototype. We have decided not to include it in the resulting figure because of the amount of different functions and aspects to consider.

Advanced user

Originally we thought about dividing the review part in two. One for the advanced users, and one for the users of the end application. After a while we found that the advanced users only would score higher or the same as the normal user in most applications. Therefore we decided only to comment that kind of user here. The advanced user would enjoy TMtab and Omnigator much more than an ordinary user. Naked Objects is possible for anyone to learn, but it took some time because of some of the new ideas they presented which compromised the ordinary way Microsoft Windows does.

4.3.5 Summary

In general Brain got an overall better score than the other applications. One reason for this can be that they have a commercial program they sell. Many users does not hesitate to write mails back to the developers if they feel there are lacks, or improvements should be implemented, especially when they have paid for it. If the developers then know something about usability engineering, they know that such kind of feedback is in fact very valuable, and try to improve. The other applications are mainly built by private people or by organisations and the demands from other users are not as strict. Developers of open source applications welcome new suggestions, and other feedback, but they have no demand or responsibility to fix it to the same extent as developers of Brain, who charge for their application. In the next chapter we will use some of the aspects from this review to form a kind of “ideal solution”. The ideal solution will be compared later on with the prototype built.

5 The Ideal Solution

Different applications have been reviewed to reveal what they can offer. Results from the reviews will be used in this chapter combined with the case from InterMedium. Inputs from our supervisor, InterMedium, and the reviews are used to explain what we consider to be the ideal solution. Describing the ideal solution will be done using the same structure as in the reviews.

In some occasions there are several correct answers to one question. These are described in this chapter and later on the prototype chapter shows the solution that knowledge-workers at InterMedium considered being the best for the InterMedium case.

InterMedium wants to describe different ontologies and an easy way to export them into an application. Navigation and editing can then be done by the end users in the model based on the ontology. As in all applications, end users are not necessarily familiar to technology. One must therefore consider the very important usability aspects discussed in the usability chapter 2.3. Technology must be hidden to the end user and only provide functionality. Technology issues will not be considered in the Ideal solution. Concepts and solutions for how InterMedium can satisfy their customers will be in focus. The Ideal Solution will be used to develop a prototype as a "proof of concept".

Ideal solution - an Utopia?

At its very best the solution will generate a faultless application with high usability based on a given ontology. The generated application will provide an environment with high usability for InterMedium's customers.

Assuming that auto-generating the application not always provides the best solution implies that some other important issues have to be taken into consideration. It is more realistic to regard the auto-generation as the first part of the work, and the second part to tailor the solution for its predefined use. The closer this generation is to the final product the less work is assigned to the developer. Since InterMedium's environment demands the possibility to make applications for different ontologies we will consider how to auto-generate applications so the developers can make a product for the end user.

To be accurate, this is not the absolute Ideal solution, even though the chapter boldly is named after it. It is merely our interpretation of what the ideal solution could be

5.1 The usage of the application

5.1.1 The User group

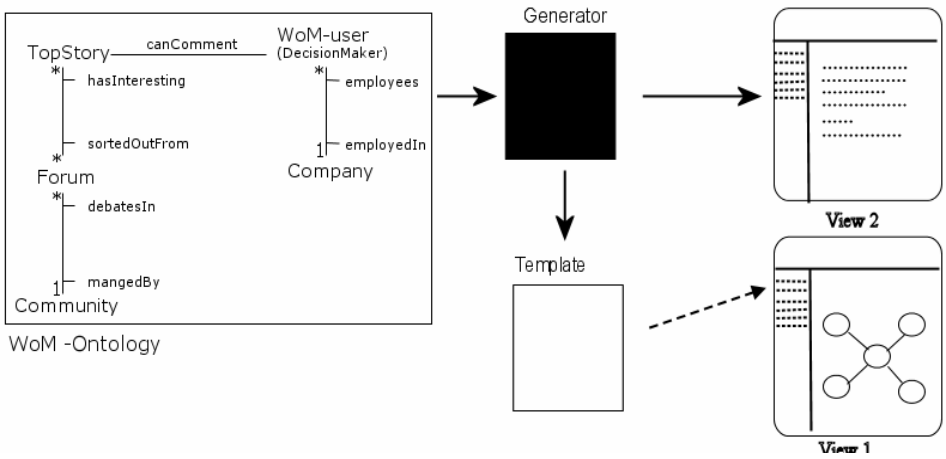
The user group is still the same as explained in the review of existing visualization applications. The ideal solution has to consider the developer building and applying the ontology he wants to use. The InterMedium user does not care how the application is developed. He is interested in extracting as much as possible of the knowledge InterMedium can provide. He also wants to, if necessary, make comments, insert new or edit existing information.

5.1.2 The purpose of the application

Using the InterMedium case in the application

The user interfaces will be generated from a given ontology. All the various ways ontologies can be described gives many challenges. A view can strongly depend on the ontology and the user's predefined requirements. As mentioned earlier, the Ideal Product will generate a view with high usability suited for the user's requirement. Figur 28 illustrates an approach to how an application can import and auto-generate a user interface based on that ontology.

View and edit capabilities in the auto generated application must be provided because of the demands given. So the developers can be able to make a final product for the end user it could be suitable to generate templates. The developer of the system can then use these templates to give high utility to the layout.



Figur 28: WoM ontology from InterMedium (explained in the case chapter) is here applied to the ideal solution. The ideal solution then generates a view making it possible to interact.

The users should be able to view and edit the information provided by the ontology. View 1 and view 2 [Figur 28] illustrate different but common ways to explore repositories. Human nature often tends to understand figures, illustrations and maps with subjects and its associations better than plain text. This especially applies when associations give much information in itself(s. As discussed in the review of TMNav, the graph presentation gave a good overview of associations and occurrences for a topic. Omnigator use text-presentation, and it was a strong solution for large maps. However the Omnigator used very many Topic Map related concepts in its representation. Most people, and especially customers of InterMedium, are unfamiliar to Topic Maps, and using this technology in the presentation should then be avoided. Figure representation, like those illustrated in TMNav, can be too comprehensive if they contain large amounts of information. Visualisation with graphic maps (se visualizing chapter, globe presentation 2.2) looses then some of its edge over a textual presentation.

5.2 The Ideal solution – criteria

5.2.1 Usability

Usability is important for many reasons as mentioned in the review- and the usability chapter. InterMedium's customers might not get all the knowledge if the usability is on a low level. It is also important for InterMedium that the customers are satisfied with the usability in the application. InterMedium will then keep customers buying their products (System acceptability chapter 2.3, [6]). The information layer and the resource level can be as optimal as possible, but at last it is the knowledge extraction that is important for the end user. The level of knowledge extraction depends much on the usability implemented in the application [6].

Learn ability

Time-to-learn, entry level and intuitiveness are keywords applied to learn ability. InterMedium's customers have to recognize the functions in the application. A good example of recognition was Brain. It is easy to use Brain for people familiar to computers. Brain has focused on novice and casual users. Programs focused at these users have a faster learning curve [6]. Brain has a fast learning curve because of its few functions implemented and a well built-in guide. The map is presented at once, so the extraction information from the model can start immediately. How to edit and add information is also very easy to understand. The functions implemented are also easy to understand for non-advanced users, like those defined to use the WoM case.

The old term "Keep It Simple"(KIS) is applicable for learn ability. Too many Features can confuse rather then help the users. Customers of InterMedium that have been away for a while, and then returns, should get into the knowledge extraction as fast as possible. They should not be distracted with many features. Simplicity must therefore be applied to our solution.

Efficiency

The customers using WoM or any other projects should feel like they operate in an effective application. Navigation must be effective in the application. In order for the user to continue using the application he has to feel that these operations are effective. If the application is large, with many elements, there should be implemented a search engine. Time is valuable, and it is not the knowledge searching but the extracting and use of it that is important. An example of this is searching in the Omnigator with the query language Tolog. Tolog is highly efficient but our user group is not capable to use it.

5.2.2 Navigation

Browse

Much good information about browsing a map came from the reviews. Presenting information with figures and drawings similar to Brain and TMNav seemed at first like the obvious solution. After analysing the reviews, this graphical interface was no longer the obvious solution. It appeared that a really large map could be rather over-complex using the same visualisation method as Brain. This occurred especially when the bidirectional associations had information and was drawn between two instances. In fact, Omnigator and TMtab were more suited for our case if you ignored the extra technology-information these had to offer. This is because it exist many top stories and communities with descriptions. Using Brain's visualisation method could then overwhelm the user. A text based presentation would be more general and has therefore higher utility for our users. A stripped version of Omnigator, visualizing maps with text and listing out the information, with no references to the underlying technology is therefore more suited for our users. Good navigation can be accomplished using information from the ontology to enhance the understanding, like TMtab and most of the graphical user interfaces. Words from the ontology can be used to describe relations instead of using specific Topic Map concepts like "association" and "type". Using words seems natural, and it is also more natural and understandable.

On the other hand InterMedium could have a large ontology with many classes and few instances of every class. In this case, it seemed like illustrations with drawings had an advantage. It is known that humans recognize,and remember a picture of a map better then a list of text. So in some cases it could be better with a graphical approach with figures and drawings to enhance the browsing, and increase the user's ability to remember the map.

Giving the case from InterMedium, a textual visualization without using drawings to illustrate associations will provide the best result. In the WoM case figures and graphs can be used to present the trends of the users.

Searching capabilities

Brain had the best score when it came to searching. Searching was done in three ways. Simplest of these searches was the instance-searcher in your map. This search was available at all time. Information you did not directly remember could be found with the advanced search function. This search was efficient and easy to use. Most of the time you know what you are searching for and then it is fast to type the word and find the wanted information in the map. The Brain's first method of search in these cases was an instance-name searcher. The InterMedium case is more content based; therefore the instance searcher is not really that necessary in our ideal solution. The content searcher on the other hand could be nice to have. It is also important to have in mind that the most important searches done in the WoM case is done by InterMedium robots, and the results are placed in the TopStory category in the map. The search engine preferred would then be the advanced one; to find information the robots has not been instructed to search for.

Model overview

None of the programs reviewed had a special built model overview. A model overview could be great when facing a large map. The most important aspects of the map are then pointed out, to enhance your ability to get knowledge from the map. The InterMedium case could benefit from an overview.

5.2.3 Edit

The InterMedium case states that editing is important. But how is a good editing done? This chapter will explain the basic editing elements and how this should be available for the user.

Insert new instances, Modify instances and Delete instances

The programs that had ability to make modifications in runtime, TMtab, Brain, and naked objects, had very different ways of doing it. Brain offered no consistency check of what you were allowed to add, so you could add whatever you found important to add, wherever you wanted to. You could also with this approach add information violating the ontology, for instance assigning wrong attributes to a person. TMtab offered a more strict set of rules, forcing you to only choose among valid attributes. TMtab also allowed you to change the set of rules at anytime, making it possible also here to end up performing disallowed insertions. The ideal solution restricts the user from inserting instances at wrong places, or assigning wrong attributes to instances. The WoM case has the possibility to add, modify comments and descriptions. What is allowed to add, edit or delete, should be defined in the ontology. Giving constraints are very important to keep the consistency.

Scripting

The scripting language should make it easy to insert many new elements and modify those. LTM[7] is a good example on how one can make a Topic Map with relatively easy syntax. When using LTM you must be familiar to the Topic Map concepts. The ideal solution should make it possible for users without any knowledge of the Topic Map concepts to script their own files. It should appear errors if the script file tries to do something conflicting with the ontology. The script file should use understandable commands as make, add and change so the user does not know the backend technology.

5.2.4 Import/Export

Considering the WoM case, exporting is not important to the customers of InterMedium. The ontology is used to make applications. Some cases demand downloading information, export functions would then offer this. If export is to be made, the exported data must be exported in a standardized format.

5.2.5 Design and Layout

Representation of a model

A model can be presented in many ways. The most important issues, no matter what presentation technique selected, is that the model is understandable. Some models demand a graphic approach with illustrations, and some models can very well give meaningful information without illustrations. Brain did a very good job presenting the information. The model inside Brain tends to be large, and "homemade" (without a strict ontology to follow). "Parents" in the model with many "children" are visualized in rows, and columns, with a scrollbar. Depending on what screen resolution you have, the more children are in the view. Holding your mouse pointer over one of the children, the property window automatically updates itself. Omnigator solved this by listing up children. To get the information about one child you had to follow the link. TMtab for protégé listed up the children, depending on what view you had. Either a new view with the information about the children appeared in a new window, or in the window you were in. The most time effective of these solutions was Brain's, because you did not have to do "go back" for each new child. Fast extracting of knowledge is valued, not because InterMedium's customers necessarily are in a time rush, but because of enhanced user friendliness. The cases from InterMedium also here put some strings to what choice is best, along with the selection of technology platform.

Illustrations

Illustrations are used to point out something important, for instance a shortcut to a function. The illustrations used have to be understandable. Too many illustrations can be unnecessary, and too few can give less functionality to the application. None of the programs except Brain had any special illustrations to picture functions, and Brain only had a couple. Depending on the specific case, and very much of what kind of user the case has, these functions have to be decided.

Available core information

Core information was in the review defined to be application specific. Cases about knowledge retrieving should not look like an online bank, or a music player. The design of the application should follow the case in some degree.

Application specific

Here we will go into some issues not directly affected by InterMedium customers, but in stead the programmers and developers at InterMedium. Each case InterMedium decides to use can be different. The building bricks used to create the browsing environment for the end user can benefit from not accessing the elements from the ontology directly. They can instead use understandable methods also generated from the ontology. Usability for the end user will not change, but the developer will appreciate the methods generated, and he can then be more efficient in the development. A good example is UML tools of today, they generate classes with attributes and methods based on a model. The generated classes do not necessarily include all functionality at first, but gives you a head start, and it could possibly save some time. It is important not to forget that the end user is much more important than the developer, and that the end user at last judges the application.

Making programs that can give a coherent generation of what is mentioned above is very demanding and time-consuming. Different approaches to analyze the application should be implemented, and real time testing by different users should be carried through. This could be done for instance using usability heuristics [9] to cover as many errors as possible, and weaknesses after the prototype has been released.

6 The Prototype

Applications are generally more and more developed using different components. Much of our time would have been spent handling the Topic Map concepts if we were to start from scratch. Therefore we use a Topic Map engine in the building of our prototype. We chose to use "Topic Map for Java "(TM4J) since it is already used by InterMedium and since it is the leading open-source project within the Topic Map.

The prototype will be a proof of concept of how it is possible to view and edit in model driven applications based on Topic Maps. For developers to easily customize applications we will make an application that generates a reflective API[14] based on the ontology. We call this API, "Model Interface" and programmers can use this component to make their own application.

"At its very best the solution will generate a faultless application with high usability based on a given ontology." – [ideal solution]. The prototype will generate templates that can be used to tailor the solution into an application with high usability. Since ontologies can differ in their use and purpose it is challenging to generate an application considering the end users needs. And to the end it is this user the program is made for. We will generate user interfaces with editing and navigation possibilities. These interfaces have templates so the user can customize them.

The following chapters explain how we chose to generate user-interfaces with editing possibilities from different ontologies. We will then see how we can tailor these generated user-interfaces into an application with usability aspects. To do this we will see how the cases from InterMedium are visualised. The results from the prototype will be presented in the Result chapter.

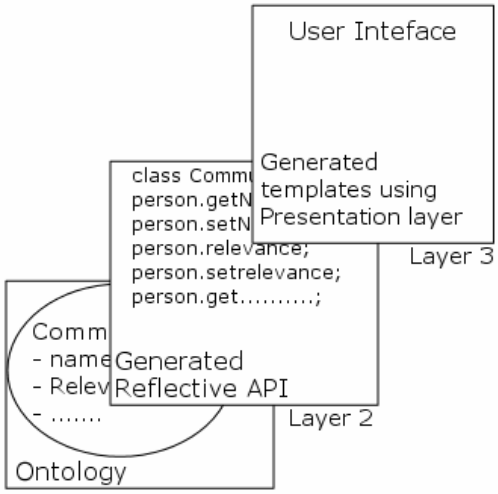
[14] Reflective API – An abstraction level used to reflect, or offer a modell with a API

6.1 The development

The development of this prototype is based on the ideal solution. This prototype will be a web application. This is because InterMedium is an Internet based company and such an application would be more available for the subscribers of InterMedium's services. This can however be changed in the future. Rendering high detailed models on the Internet takes up a lot of time. Good 3D visualising can be better in some applications. In such cases an application based on OpenGL or SWING will be better. This is why we have decided to develop the prototype with different layers.

6.1.1 Three layer model

We have chosen to make a three layer model to gain the flexibility described above. Figur 29 shows that the first layer consists of the ontology described with the Topic Map. The next layer is a reflective API generated from the ontology. It is used so the developers do not need knowledge about programming the Topic Map with its structure and its concepts. The methods in this layer are used in the generation of web sites. The presentation layer provides logical functions for the programmer. The generated web sites consist of templates so we can customize the web applications to its purpose. By using the methods in the presentation layer in the generated web sites the developers can easier understand the templates. He does not need to have any more knowledge about Topic Maps than the fact that he has to define the ontology based on this.



Figur 29 The prototype is build based on a three layer model. A three layer model can benefit the developer when he wants to customize.

The final layer is the web interface assigned for the end users. This web interface is also auto generated, and it consists of templates based on the presentation layer so the developers can tailor it for its purpose. The following chapters will explain these layers in more detail.

6.1.2 The ontology

As defined earlier: the ontology consists of the basic elements and the constraints they have. Topic Maps is a well known technology for InterMedium and it is used to describe models. As we have seen TMCL is not yet finished but much work is done in this area. The Topic Map engine have used has no support for this. Due to this lack and the fact that there is no easy way to implement some of the existing TMCL projects into this engine, we have defined constrains within the Topic Map model. By doing this the reader can also see the constraints in the model. We will use the same limitations that are described in the Topic Map chapter.

Challenges

As we have seen the Topic Map standard is so wide that you can define your map in a variety of ways. This is a great challenge for the modeller. We also have to consider that our application should be auto generated based on the ontology and therefore we cannot mix different concepts with different meaning. This should of course be avoided in general, but it is extra important when auto-generating. As we have seen there are many different concepts used in Topic Maps. Within our time scale it is difficult to implement a system working for all the concepts. This is also considered when modelling the ontology.

Modelling the ontology with Topic Map concepts

The modeller can make the ontology in very different ways using Topic Map concepts. In the InterMedium cases we have chosen to use some of the concepts within the Topic Map standards. The constraints and the types in the ontology are defined using occurrences for topics. The occurrence data says what multiplicity it has. * means you can have many and 1 says you must have one. Our prototype is limited to these multiplicities. The occurrences also say what type the topic is. n association is scoped to the name of the association, a scope is scoped to a help topic "" and an occurrence is scoped to occurrenceScope.

To indicate that a topic is editable we use a topic "editable" that we scope to the editable topics. In the business view we say that the topics are of the topic type buisnessView if they shall be at the main site. The business view is used to insert new topics and these topics appear at the first site.

6.1.3 Presentation layer

The presentation layer is a reflective API for the ontology. The purpose of this layer is to let the developer access the ontology using logical function calls. The presentation layer wraps the Topic Map concepts into more logical functions. Since this layer does not use Topic Map concepts the users do not need to know how to program the Topic Map. As seen in the previous chapter we have defined community and forum as topics in our ontology. And the communities are connected to forums with associations. They also have data like "number of users" and "language" defined with the occurrence concept. To get this data from the Topic Map the developer has to know how occurrences are connected to a topic and how to get the associations for a topic. The user needs knowledge of the Topic Map concepts and the structure to use the data in the end application. By wrapping all the Topic Map calls into logical functions, the user can use the methods without really using the Topic Map concepts. He can then call methods like getNumberOfUsers() or getLanguage() and he will get the information returned. This is why a presentation layer helps the developer.

Challenges

The presentation layer is often tailored for a specific purpose. In our case it is tailored for the ontology concepts described above. As mentioned the Topic Map structure can vary from modeller to modeller. Some might use associations to connect information to one another; others might use occurrences with or without scope. How to get and set the information can be different from one Topic Map to another, even though it somehow has the same ontology. We have to make a general implementation for this wrapped into methods so the developers can relate to this reflective API.

Generating the presentation layer from the ontology

Generating the presentation layer we have made an application called the ModelInterfaceGenerator to do this. This application uses velocity templates[15] to generate the presentation layer. There are several different ways to do code generation but the velocity templates are simple to use for proving the concept.

There are mainly two areas in focus when generating a presentation layer like this. One is to make an overview class where you can get all the different items defined in the ontology and the other to make one class for each item so you can get more information about this. We will explain how we generate these classes from the ontology and how this can be used. Documentation of how this is implemented in the ModelInterfaceGenerator application is found in Appendix C.

The Overview class

The presentation layer has to have one class for the overview of the ontology and one class for each topic. The overview provides the developer with methods where he can get and add instances from the Topic Map. The user should be able to get the instances and insert new ones if it is allowed. Here you can see the overview class generated for the ontology described above:

```
//This method returns a list of the forums that exist in the topic map
public List getForums()

//This method returns forum from a given id
public List getForumById()

//This method returns a list of the topstories in the topic map
public List getTopStorys()

//This method returns a Top story from a given id
public List getTopStoryId()
```

Figure 30: The auto-generated methods used to get information from the topic map are considerably simpler and intuitive than the ones provided by TM4J.

Item Classes

Each topic has its own class. This class has the methods for getting all the information connected to this topic. To describe this we will give an example of the community topic above. Libraries can be viewed in Appendix C.

The limitations

To make a robust and well defined presentation layer for all the Topic Map concepts are time consuming. We have earlier described how we modelled the cases from InterMedium. In this prototype we have focused on these concepts. We have not implemented methods that handle:
Adjectives given to associations, one way associations, associations with equal roleplayers, associations, connected together with equal topics, and reification.

Our prototype is only showing the most important concepts, therefore the above mentioned aspects are not included. The models of the cases from InterMedium do not include the concepts from the list mentioned above. Even though it is InterMedium's wishes to describe different ontologies we have limited the prototype to handle only these.

[15] Velocity Templates – A template tool often used in auto-generation. [14]

6.1.4 The web interface

Introduction

The web interface is what the end users will meet. As we have seen in the reviews, there are some usability concepts that must be considered when making such an interface. As we have seen in the ideal solution, InterMedium wants an easy way to develop these web interfaces from a given ontology. At the best an application could generate the user-interfaces. Automatically generated interfaces are often very general and it is difficult if not impossible to make this user friendly for the end user fit for the purpose of the ontology. To solve this we will auto-generate the web interfaces with templates so it is possible to customize them to its use.

Challenges

As mentioned in the "Ideal Solution" chapter it is almost impossible to generate user-interfaces fit for its purpose with a good usability. Our challenge is to generate them so close to this as possible and generate editable templates so it is possible to tailor the web interface for its purpose.

Generating the web interface

This generator will generate templates for each purpose so the developer can tailor each site to its purpose. InterMedium has different ontologies for different clients. A web application generator like this will do most of the work, and the programmers can use the templates.

As mentioned above the presentation layer is used to give logic to the templates.

The web site generator will consider usability. But it is depended on how the ontology is defined. How are the scope, association or occurrence concepts used? The generator will generate applications that list associations, occurrences, types and scopes in a specific order. The ontology is described with a Topic Map. Topic Maps are used to describe how resources are associated to each other and of course the information that lies within. As Figure 31 describes the model interface is used to generate two views; classic user-interface and business user-interface. Like generating the model interface we use velocity templates to generate the finished application.

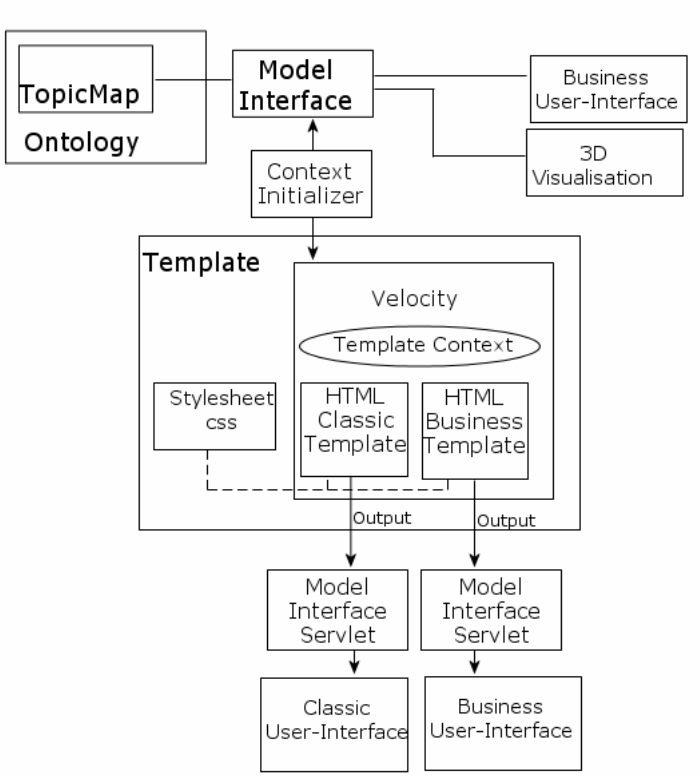


Figure 31 It is possible to integrate the model interface in other applications. In the future we may use 3D visualization methods as the most popular way of presenting knowledge information. The model interface can be imported as a component in other projects.

Classic user-interface

The classic view described is not so unlike the approach Ontopia and TM4Web [10] choose. One difference is that our library will not focus on the Topic Map concepts.

Business user-interface

The business view is used for more editing. This is somewhat similar to the models used in naked objects. But the approach is different. Editing and visualizing is done with text. This model is suitable for models where editing is important.

As we have seen illustrations can often be used to describe data. It can be a pencil for editing or a picture of a person to describe a user. Icons and other illustrations so the user can view the information will be embedded in the model interface so it will be more user-friendly. We will set default icons for this, but other users can customize this by changing it for their needs.

As Figure 31 shows the model interface is put into a context. This is because we have made velocity templates for the HTML documents, and these templates use the context. Using a servlet we will then be able to display the web application.

6.1.5 Using the Model Interface in other contexts

One of the reasons why we generated a reflective API is so it can be used in other context. It is not certain that InterMedium wants a web interface. They might want to make a 3D application visualizing the application, as Figure 31 in the visualising chapter shows. New techniques to visualizing and making the application user friendly for other purposes might be waiting around the corner. Our reviews state that text presentation is a good way, but as we mention in the visualizing chapter the development within different application differs. As Figure 31 shows the Model Interface can be used in OpenGL 3D applications or SWING applications.

6.2 Customizing the prototype

6.2.1 Customizing the content

The generated templates do not at first necessarily provide the user with the content he wants. It is therefore often necessary to customize the content. We will now take a look at how we can customize the content in the main site, the sites generated from the templates and the menu.

The main site

The main site has not any content directly from the model. It consists of the structure that the rest of the web application with a top image and a menu. Using the Overview class in the reflective API the developers can get the categories they want to display.

The sites generated from the menu

All categories have their own template file. The developers can then customize the templates of interest. The methods from the reflective API are also used here.

The menu

The menu is in a separate template file – menu.vm. This file comes with a home link. The most interesting subjects can be put into this menu. When generating and customizing the WoM case we chose to make a drop down menu displaying Top Stories and Communities.

6.2.2 Customizing the Design and Layout

The design and layout for the auto generated templates are determined using HTML and CSS[16]. The details and the relations have their own class in the CSS file. The developers can customize the CSS file and place the details and relations as they wish. Changes will apply on all generated template files. If they want to make other changes they have to customize the templates file manually.

[16] Cascading Style Sheets – Used to abstract, and simplify design and layout of larger HTML sites.

7 The prototype vs. the ideal solution

This chapter will compare the ideal solution with the prototype. We will use the structure from the ideal chapter, when discussing the difference between the ideal solution and the prototype. Early in the ideal solution we stated that it would be hard, if possible, to create an ideal solution. Relying heavily on auto-generation does not improve the usability aspect either. The prototype we compare to the ideal solution is a modification of the auto generated application. Usability is enhanced compared to a raw output from the auto-generation. The case from InterMedium is applied to the auto-generation with enhanced usability. Utility is high, since this application is fitted to knowledge retrieving for InterMedium customers only, which it is supposed to do. The same structure, at a higher level of abstraction, are used here as in the review and ideal product chapter.

7.1 Comparing criteria results

7.1.1 Usability

The users for this application, knows the general principles behind the application. Assuming InterMedium's customers have some knowledge about computers, and that they know what they have subscribe, implies that they already has climbed some steps on the learn ability curve. They know about the concepts, therefore they get a head start when learning the application. Learn ability is very important when it comes to satisfy the customers quickly. The prototype has relatively high learn ability for a user knowing the concepts in the application. Users without this knowledge will relatively quickly understand what the application is about. Using only descriptive words in the ontology gives a higher learn ability curve for all users because of the familiarity that meets them.

It is hard to give a "qualified guess" on the efficiency of our prototype. Measurements of how long time a set of actions would take should be performed. The efficiency is however, not that important if the intention is to keep the users for a long time. The time is of less importance for InterMedium customers. Simplicity and the ability to extract information in an easy way for InterMedium's customers are also important.

Our application does not fulfil the ideal case after the auto-generation based on the ontology. Special customization must take place to add icons, pictures, and functions that has not been auto generated. Luckily the reflective API provides an easy approach for the developer to enhance the usability of the application. Special customization is necessary, with or without auto-generation. This is because every customer has different logos and need special functions, special requests and other demands.

7.1.2 Navigation

Browsing in the application is kept simple. All auto generated words are basically taken from the ontology. Well designed ontologies will then provide better understanding of the browsing environments than other ontologies. The generated framework is also kept simple. The developers can and should add functionality to enhance the browsing and give more options. This is a question of priority and investigation of the users needs. All users do not demand more functionality than provided by the auto generated application, even though it is assumed to rarely be the case. Searching capabilities is not implemented in the application also to keep it simple. However tolog should be used if there is need for a search engine. A model overview must be customized by default.

7.1.3 Edit

Insertion and editing of information is done by clicking the edit icon. The edit icon is the same for all edit options. Using the same icon consequently in these operations will help the user recognize what kind of operations is allowed on items in the application. If needed, the icon can easily be changed. The prototype does not support export options, and it is not possible to import the ontology more than once, before the auto-generation of the application. Import and export can easily be implemented in the application since TM4J has support for this. We did not consider this important for the prototype since its purpose is to show the possibilities auto-generation gives.

7.1.4 Design and layout

The model is presented in a text based browsing environment, which we also considered as the best solution[see the Ideal solution chapter 5] to use with the InterMedium case. A simple design was produced using only the necessary information from the ontology. All communities and Top Stories are available from a menu. This is not auto generated in the model overview. The illustrations we used in this demonstration are all fictive, but realistic compared to what a real application would show to InterMedium customers.

7.2 Generation of the user-interface

Ideally the ontology could be imported and a user interface would automatically be generated. Our solution import ontologies described using Topic Map concepts. The modellers have to use these concepts [see appendix C] in a certain way to be able to use our generator. The output is not a perfect user-interface but it has templates the developers can customise to its need.

Ideally it would be possible to auto-generate an application based on the ontology and modify the application to some specific use. Discovering that the ontology needs more elements demands a new auto-generation. Modifications done to the application after the first application should then not be changed after the second auto-generation. This is not implemented in our prototype.

8 Result

The previous chapters have given the basis for our research within this area. This chapter will first present the results we found important from the reviews. Then the prototype shows the results.

8.1 The Reviews

The reviews gave us valuable information on how other projects have implemented usability engineering, how they visualize and present information, and how their design and layout was. All the applications had their own strengths and weaknesses. The applications we reviewed also had slightly different user groups. Focus and the utility of the applications therefore differed. The application must in as high grade as possible fit the user group. The theory Jakob Nielsen presents [6], says that it is important with high usability in an application. The reviews only emphasised what the theory states. If for instance two novice users were to choose among Brain and TMtab to browse a model, they probably would choose the commercial Brain. This is because of Brains solutions had high learn ability, higher utility for extracting knowledge, and rather high efficiency. To sum up Brain got a general better score for the novice user. TMtab is more focused on the expert users, as an ontology building tool, and had many advanced functions available at all time, making it over-complex, just to be used as and knowledge extracting tool. The “Keep It Simple” principle seemed like a very good rule of thumb to use when implementing, especially when considering the user group presented in the InterMedium case.

8.2 Auto generating the reflective API

First the reflective API is generated based on the ontology. The methods used in this API are generated from the topics in the Topic Map. A class for each topic is generated and these classes contain methods for getting the information related to this topic. Methods for getting the base name, the occurrences and the associations are generated. The constraints we have defined in the Topic Map, help to determine whether these methods should return a list or the specific instance. If it is possible to have many instances, one method that returns a list of them are generated and one method that returns the instance based on the id is generated. Also an overview class is generated. This class has methods so the topics in the Topic Map can be retrieved.

8.3 Generating the Web Application

After the reflective API is generated, the prototype generates a web application with customisable templates using the reflective API. You can either generate a classic view or a business view (see chapter 6). This chapter will show the generated applications and show how we chose to customise the WoM case displayed with the classic view.

The functionality of the generated web application

The prototype uses text navigation. The auto-generated application uses a reflective API to access the information in the model. The information in the model that relates to other information is hyper linked to eachother. Figure 32 shows that the forums that the top story is sorted out from are hyperlinks. The details are not hyper links.

Figure 32 The auto generated page showing TopStory.

Editable data appears with input fields. Text areas are used to insert details. This can for instance be a description. To insert the information the reflective API is used. When the users insert a new association a drop down list with the allowed associations is displayed. The auto-generated drop down lists does not check if the associations are already made. This must be done manually.

An occurrence with a scope has a text field to insert the data of the occurrence and a drop down list where you can choose the allowed topics you can scope. Comments to top stories are a good example here.

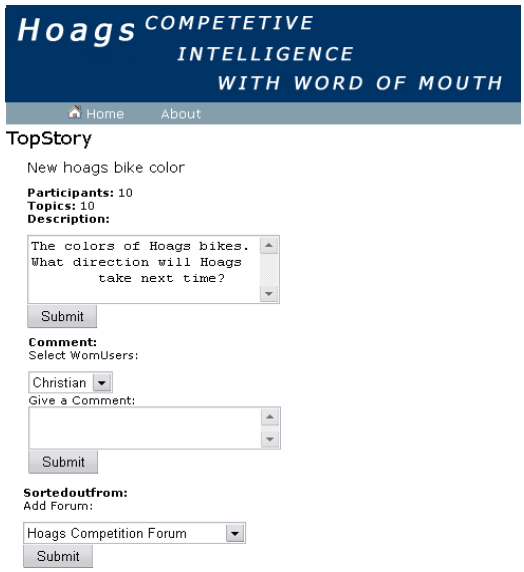
When users insert a new topic the available information appears. The user can then insert it.

Figure 33 Icon used as shortcut to the overview page. “home”

It is also generated a menu at the top. This has a home icon (see Figure 33) that returns you to the first page. This page is generated empty and must be specialised to its need. The menu is also meant to be specialized to its needs. Important information is placed here.

Specific functionality for the classic view

The classic view represents a model. It uses a XTM file that defines the ontology to generate this view. If you have stored more information than the ontology, you can use this file when the application loads. This file has to be called model.xtm.



Figur 34 TopStory from the WoM ontology

Specific functionality for the business view

Unlike the classic view the business view can generate a main site with content. This is defined in the Topic Map model as described in chapter 6. Figure 35 shows the auto-generated main site for the business view. This site provides editing possibilities. But it does not check it the text-field has content. We have made customised this into out generated application. The main site provides editing possibilities for the name of the topic and the occurrences. The associations can be added after you have entered this information.

Figure 35 The business view generated when using the News ontology.

Customising the classic view

As mentioned the reflective API is used to access the Topic Map. The developers can use this API to get or insert the information they want. They also use logical names dependent of the ontology.

The prototype generates velocity templates used in the application. These templates are using the reflective API so it is easy to understand how the data is displayed. Figure 36 shows how the top stories are extracted and displayed using html. In this case you get several top stories and the list is iterated. A top story can only have one description and then it will be displayed as Figure 37 shows.

```
#foreach($obj in   $topstory.getComments() )
<br>
$obj.getCommentText()
[womUsers: $obj.getWomUsers().getWomUsersText()]
<br>
#end
```

Figure 36 The code to extract comments on a specific topstory.

```
<b>Description:</b>
<br>
<br>
$topstory.getDescription().getDescriptionText()
```

Figure 37 The application is easily customized with these methods. Compared to the methods one usually use, these are very simple and straightforward. Using the documentation and the ontology is enough to use all

To change the design and layout you can either change the html and velocity code directly or you can use a generated cascading style sheet file. This has classes for links, details and the associations between the information.

The menu are in an own file. This file is included in the other templates so the user can act only at this file to make the menu. Also the top picture is ment to make for the special case. But the layout can be customised so it does not need to have a top picture at all.

Figur 38 The customized web application for classic view.

Deploying the application

We have used an ant[17] file to deploy the web application. All the code is generated and by running the ant file you get a standalone web application you can deploy at the Tomcat server.

8.4 Assigning usability to the final application

Two ontologies were tested on the prototype. The WoM case was customised to its used.

The overview should display the relevant data. In the WoM case top stories are the most important, and therefore listed at the main page. Trends for a product should be displayed with graphs or similar figures. We made dummy figures to illustrate how the trends for a product can be used.

Figure 39 Here is an example showing all available forums allowed to submit.

Icons are used to give more meaning to the different information. Top stories and communities got their own icons, since these are the most important aspects. Instead of showing all the editable information in text areas we have used icons that you can press to start editing the information. Figure 39 shows the drop down list that appears after we have pressed the insert sorted out from association. Except comments since we consider it important to have the possibility to enter a comment.

A menu is used to easy access the most important topics easily.

Figure 40 The ontology gives users the possibility to make comments, marked with their name. Here is the auto generated form and dropdown. Customizing is easy

Error messages for the users should appear when a user makes a mistake. The prototype based on the WoM case, disallows the user from making mistakes by only giving him the allowable options. A forum already mentioned should not be available again.

[17] Ant file – Java based build tool [15]

9 Discussion

This thesis started to explain how to use Topic Maps to structure information, how to visualise it and what usability aspects one should consider for an application. Two purposes of this thesis were sorted out. First we were to auto generate model driven application with editing possibilities. Second these applications should be customised for users who want to extract information. Auto-generation is used since InterMedium has several different ontologies they wish to use and describe. Auto generating the application for the users will benefit InterMedium. The output from output Auto-generation often tends to be simple, general and technical oriented. It exists many different ways to make a model, therefore it is very challenging to combine usability engineering in an auto generating application.

How is it possible to simplify the work for InterMedium's developers without violating the final application for the end users? A prototype has been developed to highlight these problems. Further the prototype demonstrates our suggestions to how the auto-generation can help the developers to easily make a prototype.

This chapter will discuss:
- The reviews and Usability aspects
- Auto-generation of a knowledge base with Topic Maps
- Navigating and building the knowledge base
- Customise the application and assign usability
- Using the reflective API in other contexts

9.1 Auto-generation of a user-interface with Topic Maps

The Topic Map standard saw the day light at the year 2000, so it is a young standard. Several Topic Map engines exist, but few in a mature state. We chose TM4J because it is one of the leading Topic Map engines, even though it lacks some important features. In this chapter we will discuss how Topic Maps are suited for auto-generation of a knowledge base.

Constraints are important to use when auto generating an application. TMCL is not yet finished and TM4J has not implemented another constraint language. We had to find alternative solutions, and chose to define the constraints within the Topic Map. Modellers have to be aware of how these constraints work, in order to auto generate the knowledge base when using this method. Ideally we would have gone for another solution. Modellers can be confused by using a not standardized method. A much better solution would be to define the constraints in a schema language. Constraints are then separated from the model itself and it is easier to understand for the persons handling the Topic Map

Associations in Topic Maps are well suited to describe a knowledge base. Associations are a strong tool to describe how topics interact with each other. As we have seen in chapter 2.1 you can also tell what roles the different topics play, and assign more meaning to the associations. Bi-directional associations can use the scope concept to determine for what topic the description is valid for. To make it even clearer you can make reification on the association to tell more about it. Topic Map modellers can therefore structure their model in many different ways. Handling all these different ways of modelling is very challenging when auto generating an application. One modeller can use associations while another might make occurrences and scopes to model the same thing.

It is very demanding for the programmers and the processing of the application when one program directly against the TM4J API to get the associations. tolog is a query language for Topic Maps is designed to make the query of Topic Maps easier. Auto-generation using tolog gets easier especially when dealing with associations. A query language like tolog helps auto generating the applications based on Topic Maps. TMQL is not yet as mature as SQL, and tolog has therefore one important lack. This is editing of the Topic Map. Editing must be done using the API provided from TM4J. Combining tolog and the TM4J API in the generated templates would provide a very technology oriented interface to the model, and it would be very difficult to understand.

To solve this we made a reflective API wrapping the advanced code into more understandable object oriented methods. The developers do not need to consider the Tolog and TM4J API code when extracting and editing information from the map. This is a very scalable and understandable solution for most programmers. The API developed in this thesis does however not solve all the different modelling options you have using the Topic Map concepts.

Topic Map is a well structured standard with lots of modelling possibilities. When TMCL and TMQL are properly developed Topic Maps seemed well suited for auto generating. Even without this mechanism we have managed to define a set of rules and generate a web interface using a reflective API also generated from the Topic Map.

Since Topic Map uses the XML format XTM, it is possible to go from UML to Topic Map. Following standards like UML can make it easier for the developers to make stable Topic Maps. Developers can then sketch a solution for the client with UML and generate a Topic Map from the UML model. Our prototype makes it rather difficult to model a Topic Map. Using UML for this can make our prototype a very power full tool in the future. The developers can make the UML model with the clients and then auto generate both the reflective API and web interfaces. Since all the methods are made and the web interface has templates using this, the developers can customize it either with designers or with the clients.

9.2 Navigating and building the knowledge base

During the reviews we looked at several different solutions to browse a model. Some were text-based, some presented with figures and some were combinations of these. Most of the applications were generally made to import or build a model. Developers or modellers making models often use this kind of applications. Solutions used to help developers and modellers often uses concepts related to the technology.
InterMedium wanted to make the end application without ending up with backend technology appearing the end user application the so we were to find a method suited for the end users.

InterMedium's users want to retrieve the information InterMedium has available. Some of this information is more important than other. It is the main site that the users first see. The users have expectations they want to get satisfied. The index site should therefore provide a good extraction of the most important information the user wants, without overwhelming the them. The main site should therefore be tailored to what kind of demands the users have.

A menu including the most important information should at every time be available. Also icons for information often used can help the user to navigate. It is near to impossible for the program to determine which information that is most important. The home icon is used in almost every context so we have chosen to auto generate this in the menu. What the main site will include differs from project to project. We will in the next chapter discuss how we can customize the main site and the menu.

Building the knowledge should be easy for the users. The users should not be able to make mistakes. If this happens they should get error messages. We have chosen to hide the editing possibilities for the users editing is not allowed. The reason for this is to avoid misunderstandings. Ideally the users should not be able to make mistakes. In our model we uses the constraints to avoid this.

9.3 Customise the application and assign usability

We have chosen to generate velocity templates so the developers can assign higher utility to the application. Reflective API and html are used by the templates. A cascading style sheet is also generated so you easily can change the layout and design. If you want to extract other information from the Topic Map you can use the methods in the reflective API. Alternatively we could choose not to customize the application and describing the layout issues in the model.

The user is always right [6], and inputs from the users in the prototype development are valuable. The user knows and has some ideas about how the applications should be used. Building application with user inputs is valuable and can enhance and improve the usability and utility. However, the user is not always right[6], and do not always know what is best. Care must therefore be taken when using these user inputs. Developing the application and at the same time getting user inputs can be time saving. Naked objects often referred [11] to the time savings of development with user help. Auto-generation will make it possible to develop applications at the same time as the user is coming with inputs.

If it is possible to easily make the model and generate a starting point like we did it is easier to make the final application for the clients. You can easily change the templates since they use logical calls.

9.4 Using the reflective API in other contexts

The reflective API can be used in many other contexts. A good reflective API handling all the Topic Map concepts can help programmers to structure their information in a better and faster way. A problem with Topic Maps now is that it can be difficult to extract and manipulate information using all these concepts. But if this is done in the background and you only need to call logical methods like insertTopStory and getTopStories you can easily make your own program.

In this thesis we have looked at different ways to visualise data. The visualisation is under development with the gaming industry as the pioneers. Better processors makes it possible to make better graphics. It is not unlikely to assume that the information can be structure in much better ways then today. Even though we did not choose to use graphs, the future might come with new logical presentation making the visualisations even better. This could be 3D visualisation or even virtual reality applications using sounds and movie sketches. In that case the reflective API can be used to get and insert the information from the Topic Map. This would make it easier to render these applications.

10 Further Work

A prototype which auto-generate an application using ontologies as foundation has been developed during this thesis. The prototype has certain limitations. Thoroughly testing should be performed, before using this prototype to guaranty stability and error free operation.

More usability engineering can be implemented automatically by the auto generation. It should be possible to set more options before auto-generating the application. This can be options similar to the selection of classic or business view. Usability tests and evaluations should be performed on all auto generated functions to ensure the the selected function is the correct ones. Ideally there should be close to zero optimization to the application before using it.

When a constraint language has been standardized, it should be implemented, rather than defining the constraints directly in the Topic Map as we have done. Constraints implemented in this prototype are limited merely the most important ones. When a standard to define constraints has been made all different constraint aspects must be implemented.

The constraints should consider more features in the auto-generation based on the constraints. This could be options where you can choose between to selections. This prototype differs from zero or many associations.

Error handling could also be auto generated. If this should be auto-generated it is important to consider that the users should have the possibilities to change these. Error handling can be different depending on the situations.

The auto-generation should be easier, and it should also be easier to describe the ontology.

11 Conclusion

This thesis has evaluated existing model driven applications for building and navigating Topic Maps or similar knowledge structures. A prototype, auto-generating an application based on an ontology, has been developed. Much important information was retrieved from the reviews. One of the notable aspects was the fact that simplicity really paid of. The results from the reviews helped us to form a prototype.

Combining usability engineering and auto-generation has been the main challenge in this thesis. To see how these two parts can be combined, we had to break them apart and analyze them. Usability engineering is very important to apply to the applications if its purpose is to satisfy the end users. The end users are concerned with the outcome and do not use, or care, about the auto-generation. The auto-generation's purpose is to help the developer. One reason to use auto-generation is to speed up the development phase. Here it is the possibile to combine usability engineering and auto-generation.

Assuming that auto-generating the basis of an application is faster than building it manually gives more time for the developer to apply usability engineering. First of all, the auto generation abstracts the developer from the technology oriented Topic Map engines. Second and the most important issue that will benefit the developers, is the reflective API with more understandable methods than the API provided by the Topic Map engines. The ontology is used extensively to make these understandable methods. Third the web application is generated with customizable templates. These templates are also generated with concepts from the ontology to enhance the understanding of the templates. The reflective API is used to retrieve information from the model. The result we ended up with was the auto-generated three layer model based on the ontology.

Auto-generation of such a three layer model will however give lacks to the program in terms of speed. A manually build program will have higher performance. The error handling and making the system stable must be tested and developed over time.

Tools like UML can be used to define the model, and it can make it easier for the developers to work directly with clients. The clients can then be more involved in making the ontology and even see a user interface while developing.

We have shown that generating the three layer model can benefit the developer directly, and indirectly the end user since the developer then can spend more time with usability engineering. If UML was used to model the ontology, the developer would get even more time he could use to make the application more suited for the end users.

12 References

[0] T.Gruber
A translation Approach to portable ontology specifications
Knowledge acquisition
Vol. 5 1993. 199-220

[1] The Topic Map standard
http://www.topicmaps.org/xtm/1.0/

[2] Techquila
[http://www.techquila.com/pi_reification.html]

[3] IsoTopicMaps.org Topic Map Query language
http://www.isotopicmaps.org/tmql/

[4] IsoTopicMaps.org Topic Map Constraint language
http://www.isotopicmaps.org/tmcl/

[5] XML Topic Maps: Creating and Using Topic Maps for the Web
By Jack Park and Sam Hunting
ISBN 0-201-74960-2

[6] Usability Engineering
By Jakob Nielsen
ISBN 0-12-518406-9

[5] German Internet Magazine
http://www.internet-magazine.com/features/jakob1.asp

[6] German Stuttgarter Zeitung
http://www.stuttgarter-zeitung.de/stz/page/detail.php/228123

[7] Linear Topic Map Notation
http://www.ontopia.net/download/ltm.html

[8] Bond University Topic Map – Knowledge engineering
http://topicmaps.bond.edu.au/

[9] Usability heuristics- Jakob Nielsen
http://www.useit.com/papers/heuristic/heuristic_list.html

[10] TM4Web
http://www.tm4j.org/tm4wb-velocity.html

[11] Naked Objects
http://www.nakedobjects.org/

[12] Topic Map Query Language
http://www.ontopia.net/topicmaps/materials/tolog.html

[13] Object Management Group
http://www.omg.org/mda

[14] Velocity Templates
http://jakarta.apache.org/velocity/

[15] Ant java based compiling tool
http://ant.apache.org/